TARTU ÜLIKOOL

MATEMAATIKA-INFORMAATIKATEADUSKOND

Arvutiteaduse instituut

Infotehnoloogia eriala

Heike Randlahe

Programmeerimisalased koolikülastused

Bakalaureusetöö (6 EAP)

Juhendaja: Eno Tõnisson

TARTU 2014

Programmeerimisalased koolikülastused

Lühikokkuvõte:

Käesolev bakalaureusetöö on mõeldud eelkõige materjalina järgmiste programmeerimisalaste koolikülastuste tegijatele. Töös tuuakse välja, miks tuleks üldse programmeerimisalaseid koolikülastusi teha ning kuidas võiks nendeks ette valmistuda. Lisaks räägitakse ka programmeerimiskeeltest ning lauamängust, mida kasutasid 2013/2014. õppeaastal koolikülastuste tegijad. Samuti on antud töös välja toodud külastatud koolide õpilaste ning õpetajate tagasiside ja koolikülastajate endi muljed ning soovitused järgmistele programmeerimisalaste koolikülastuste tegijatele.

Võtmesõnad:

programmeerimisalane koolikülastus, Python, Scratch, lauamäng „*The Robots Game*“

Visiting Schools to Introduce Programming

The current bachelor’s thesis is meant as a guide for people who are going to visit schools in order to introduce programming to students. The thesis contains information about why one should visit schools to introduce programming and how to prepare for the lessons. In addition programming languages and the board game used in 2013/2014 school visits are discussed. The thesis also contains feedback and recommendations from the students, teachers and people doing the visits.

Keywords:

visiting schools to introduce programming, Python, Scratch, “The Robots Game” board game

Sisukord

Sissejuhatus

Tänapäeva kiirelt arenevas maailmas on järjest suurem osatähtsus igasugustel tehnoloogiatel, olgu selleks siis nutitelefon või firmasisene meiliserver. Järjest rohkem kasutatakse ka erinevaid sotsiaalvõrgustikke, uudisteportaale ja veebipäevikuid. Populaarsed on ka erinevad mängud nii arvutites kui ka telefonides. Inimesed kasutavad erinevaid tehnoloogilisi vahendeid, kusjuures enamik ei mõtle, kuidas need loodud on. Tõenäoliselt teatakse, et kasutatav tarkvara on kellegi poolt programmeeritud, aga kuidas käib programmeerimine ja mis see tegelikult on, jääb siiski paljudele teadmatuks.

Kuna tehnoloogia osatähtsus järjest suureneb, siis on loogiline, et kasvab ka vajadus vastavate spetsialistide jaoks. Oma ala spetsialistid sirguvad muidugi töötamise käigus, kuid koolis omandatud algteadmised on need, millele on võimalus rajada spetsialisti oskuseid. Üks võimalus algteadmiste omandamiseks on õppida informaatikat. Siiski on „informaatika“ mõiste inimestele segadust tekitav, kuna ei teata, mida see endast kujutab ja kuidas informaatikat üldse õpitakse.

Selleks, et laiendada õpilaste silmaringi informaatika valdkonnas ja võib-olla ka lihtsustada nende tulevase eriala valikut, viisime koos Aire Kuressoni ja Tauno Paltsiga läbi programmeerimist tutvustavad koolikülastused. Külastuste eesmärgiks oli lisaks õpilaste silmaringi laiendamisele ka Tartu Ülikooli informaatika eriala õppimisvõimaluste tutvustamine. Lisaks said õpilased külastuste käigus teada ka, mis on programmeerimine ning said luua koolikülastajate abiga lihtsa programmi.

Bakalaureusetöös on tihti kasutatud sõna „külastaja“, millega on mõeldud programmeerimisalase koolikülastuse läbiviijat. Lisaks on samas tähenduses kasutatud ka väljendeid töötoa läbiviija, töötoa juhendaja ja koolikülastuste tegija. Neil kõikidel väljenditel on oma tähendusvarjund, aga antud materjalis kasutame neid sünonüümidena.

Käesolev bakalaureusetöö on mõeldud eelkõige materjalina järgmistele programmeerimist ja arvutiteaduse instituuti tutvustavate koolikülastuste tegijaile. Lõputöö on koostatud 10 koolikülastuste põhjal, mis viidi läbi 2013/2014. õppeaastal arvutiteaduse instituudi poolt Hariduse ja Infotehnoloogia Sihtasutuse (HITSA) toetusel. Selle koostamisel on tuginetud programmeerimisalastele koolikülastustele, mis viidi läbi Aire Kuressoni, Tauno Paltsi ja käesoleva töö autori poolt. Lisaks on kasutatud informatsiooni, mis saadi Aire Kuressoni ja

Tauno Paltsiga läbiviidud intervjuudest ning nende poolt koostatud materjalidest. Toetutud on ka õpetajate ja õpilaste poolt antud tagasisidele.

Lõputöö raames tuuakse välja, milleks üldse koolikülastusi teha ning kuidas nendeks ette valmistuda. Tutvustatakse lühidalt programmeerimiskeeli Scratch ja Python ning selgitatakse, miks valiti just vastavad programmeerimiskeeled esmaseks tutvuseks. Samuti on toodud välja ka näidistunnid koos töölehtedega ning koolikülastuste tegijate poolsed soovitused ja ettepanekud järgmistele koolikülastuste tegijatele.

1. Koolikülastuste tegemine

Antud peatükis räägitakse lähemalt, miks peaks programmeerimisalaseid koolikülastusi tegema. Kahjuks ei suutnud töö autor leida informatsiooni sarnaste koolikülastuste kohta mujal maailmas ja seega on tuginetud informatsioonile, mis saadi läbiviidud kümne koolikülastuse käigus nii teistelt koolikülastajatelt kui ka õpilastelt ja õpetajatelt.

Peatükk on jagatud kaheks osaks, millest esimene räägib programmeerimisalase koolikülastuse läbiviimise tähtsusest nii koolikülastaja (ka ülikooli) kui ka õpilaste jaoks. Teises osas keskendutakse sellele, mida kujutab endast programmeerimisalane koolikülastus. Tuuakse välja eraldi tunni ülesehitus nii gümnaasiumis kui ka põhikoolis läbiviidava tunni jaoks. Täpsemalt on iga punkti kirjeldus ja tegevused välja toodud peatükis 2.

1.1. Koolikülastuse tähtsus

1.1.1 Tähtsus koolikülastaja seisukohast

Koolikülastuste eesmärgiks on tavaliselt mingisuguse teabe jagamine õpilastele, õpetajatele või kooli juhtkonnale. Meie poolt läbi viidud programmeerimisalased koolikülastused olid ennekõike suunatud õpilastele. Võib öelda, et koolikülastuse eesmärgiks oli laiendada õpilaste silmaringi. Tunnis viibijatele tutvustati õppimisvõimalusi Tartu Ülikooli informaatika erialal ning seletati lühidalt, mis on programmeerimine ning üldse infotehnoloogia.

Koolikülastaja Tauno Paltsi arvates on programmeerimisalaste koolikülastuste käigus võimalik tutvustada õpilastele seda, millega ülikoolis tegeletakse ning samuti saada õpilastelt vahetut tagasisidet. Õpilastega viidi läbi tund, mille jooksul paljud õpilased said luua oma esimese programmi või mängida programmeerimisega seotud lauamängu. Selle käigus oli koolikülastajatel võimalus tekitada või süvendada õpilastes huvi informaatika õppimise vastu.

1.1.2. Tähtsus õpilaste seisukohast

Võib öelda, et programmeerimisalased koolikülastused on õpilaste seisukohalt kasulikud. Õpilased saavad teada, mida kujutab endast programmeerimine ning oma esimest programmi luues näevad ka, kas neid võiks huvitada informaatika eriala. Samuti on õpilastel võimalik küsida täpsemalt sisseastumistingimuste ja õppekava kohta. Lisaks toodi õpilastele näiteid ka võimalikest töökohtadest ning selgitati täpsemalt, milliste huvidega inimesele vastav töökoht sobida võiks.

Aire Kuressoni arvates on õpilaste vaatepunktist programmeerimisalaste koolikülastuste tegemine oluline ka sellepärast, et läbiviidav tund pakub vaheldust tavapärasest koolipäevast. Lisaks on ka teema huvitavam, kuna erineb suuresti sellest, mida muidu koolides õpetatakse.

1.2. Koolikülastuste olemus

Gümnaasiumis ja põhikooli kolmandas astmes tutvustati õpilastele programmeerimiskeelt Python, põhikooli teises astmes oli programmeerimiskeeleks Scratch. Lisaks viidi läbi ka programmeerimisega seotud lauamängu „*The Robots Game*“ [1] mängimise tunde (lauamängust täpsemalt peatükis 2.2.3.). Lauamängu said mängida nii gümnaasiumi kui ka põhikooli õpilased, olenevalt külastatava kooli soovist. Kuna meie poolt külastatud koolides toimus lauamängu mängimine põhikooli klassides, siis on lauamängu tunnikava toodud välja põhikoolidele mõeldud tunni ülesehituse osas.

1.2.1. Tunnikava gümnaasiumile

Osades gümnaasiumites on õppetunni kestus üle 45 minuti. Selle tõttu peab olema ka tunni ülesehitus olema sobiv pikema tunni jaoks. Samas aga peab olema võimalik osade pikkust ka vähendada, juhul kui tegemist on siiski 45-minutilise tunniga.

Üldine tunni ülesehitus gümnaasiumile oleks järgmine:

1. Sissejuhatus
2. Õppimisvõimaluste tutvustamine
3. Programmeerimiskeskkonna tutvustamine ja näidisülesanne
4. Ülesannete lahendamine
5. Tunni lõpetamine

1.2.2. Tunnikava põhikoolile

Põhikoolile mõeldud tunnikava erineb gümnaasiumi tunni ülesehitusest selle poolest, et põhikooli tunni pikkus on 45 minutit. Samuti ei tutvustatud põhikoolile õppimisvõimalusi. Järgnevalt on välja toodud tunni ülesehituse kava programmeerimist tutvustava tunni jaoks ning samuti ka lauamängu tunni jaoks.

Programmeerimiskeelt tutvustava tunni ülesehitus:

1. Sissejuhatus
2. Programmeerimiskeskkonna tutvustamine ja näidisülesanne
3. Ülesannete lahendamine
4. Tunni lõpetamine

Lauamängu tunni ülesehitus:

1. Sissejuhatus
2. Reeglite tutvustus ja näidiskäigud
3. Mängimine
4. Kokkuvõte
5. Tunni lõpetamine

2. Programmeerimisalase külastuse läbiviimine

Selles peatükis on välja toodud, millele võiks programmeerimisalaste koolikülastuste tegemise juures tähelepanu pöörata. Peatükk on jagatud kaheks osaks, millest esimene keskendub enne programmeerimisalase koolikülastuse läbiviimist tehtavatele kodustele ettevalmistustele. Teine osa keskendub tunni läbiviimisele külastatavas koolis, kusjuures eeldatakse, et soovitatud kodused ettevalmistused on tehtud.

2.1. Kodune ettevalmistus

Programmeerimisalaste koolikülastuste tegijail tuleks silmas pidada, et antud koolikülastuste jaoks ei piisa lihtsalt kohaleminekust. Olenevalt läbiviidavast tunnist või tunniosast, kui osad on külastajate vahel jagatud, tuleks teha ettevalmistusi. Tundi on võimalik läbi viia ka ühe koolikülastuse tegija poolt. Meie puhul oli tund jagatud maksimaalselt kahe inimese vahel, kellest esimene rääkis sissejuhatava osa ning teine viis läbi programmeerimiskeele tutvustuse. Programmeerimiskeele tutvustuse osa ajal aitasid ka teised töötoa läbiviijad õpilasi tekkinud küsimuste ja probleemide korral. Järgnevates alapunktides on toodud välja soovituslikud ettevalmistused erinevate tunni osade kohta.

2.1.1. Sissejuhatus

Koolikülastajal, kes viib läbi sissejuhatuse, oleks soovituslik kasutada slaide. Slaidide kasutamine võimaldab teksti mitte päheõppimist ning samuti on kuulajatel lihtsam jälgida, millisest teemast hetkel jutt on. Slaididel ei tohiks olla kogu räägitav tekst, vaid ainult tähtsam informatsioon, millele tugineda. Sissejuhatuse jooksul ja ka lõpus võiks anda õpilastele võimaluse küsimuste esitamiseks.

Soovituslik oleks ka vähemalt korra proovida esitlust läbi teha. Nii saab sissejuhatuse tegija kontrollida, kaua tal sissejuhatuse tegemise jaoks aega kulub. Vajalik oleks ka teada, millisele vanuseastmele sissejuhatus tehakse ning vastavalt sellele valida, millest rääkida. Vastavalt kuulajaskonna vanusele, võiks sissejuhatuse tegija otsustada, kas ja kui pikalt räägitakse õppimisvõimalustest ja sisseastumistingimustest.

Lisaks tuleks arvestada ka tunni pikkusega, mida pikem on läbiviidav tund, seda pikem võib olla ka sissejuhatus. 45-minutilise tunni sissejuhatus võiks olla viis kuni kümme minutit, et õpilastel jääks piisavalt aega programmeerimiseks või lauamängu mängimiseks.

2.1.2. Programmeerimiskeele tutvustus

Programmeerimise osa peaks olema kõikidele õpilastele jõukohane ehk ei tasu lasta kohe midagi väga keerulist teha. Kuna eesmärgiks on ikkagi programmeerimist tutvustada ja õpilastes huvi tekitada, tuleks ülesandeid valides arvestada, et õpilastel oleks võimalus iseseisvalt midagi valmis teha. Kindlasti tuleb mingit osa seletada ja ette näidata.

Ülesanded tuleks valida nii, et tutvustatud käskluste abil oleks võimalik neid lahendada. Mugavuse mõttes võiksid õpilaste jaoks olla ettevalmistatud ka töölehed, kus on olemas lahendatavad ülesanded ning käsklused. Samuti võiks olla töölehel mõned keerulisemad lisaülesanded, kuna võib esineda olukordi, kus mõni tunnis olev õpilane on juba hetkel tutvustatava programmeerimiskeelega tuttav ning antud ülesanded on tema jaoks lihtsad või saavad ülesanded kiiresti lahendatud.

Programmeerimisülesanded ei pea kindlasti olema ise koostatud. Antud töö lõpus on toodud ära ka Aire Kuressoni ja Tauno Paltsi poolt koostatud töölehed (vt lisa 1, 2). Olenemata sellest, kas ülesanded on ise koostatud või mitte, oleks soovituslik nad ikkagi vähemalt korra enne läbi lahendada. Näiteks koostas Aire Kuresson enda poolt läbiviidud tunni ProgeTiigri [2] veebilehel olevate Tauno Paltsi poolt koostatud videote alusel. Paaris esimeses koolikülastuses kasutas Aire töölehti, hiljem andis ta õpilastele ülesandeid jupi kaupa. Ta lahendas ülesanded ise läbi ja pani kirja orienteeruva ajakava (vt lisa 3, 4), milles arvestas, kui kaua peaks õpilastele aega andma, et nad saaksid vastava ülesande tehtud.

2.1.3. Lauamängu mängimine

Programmeerimisalastel koolikülastustel kasutati lauamängu, kuna alati ei ole võimalik teha tundi arvutiklassis. Lauamängu „*The Robots Game*“ [1] peale sattus Tauno Palts, kui otsis informatsiooni programmeerimisega seotud mängude kohta. Vastav mäng osutus valituks, kuna seda saab seostada programmeerimisega (vt lisa 5). Samuti saadi lauamängu autorilt luba mängu kasutada.

Lauamängu osa läbiviimiseks on kindlasti vajalik teada reegleid. Meie poolt kasutatud lauamängu „*The Robots Game*“ reeglid näivad esmapilgul üsna keerulised (vt lisa 6). Selleks, et reegleid õpilastele paremini seletada, koostas Aire Kuresson slaidiesitluse, milles tõi välja näidiskäigud. Samuti koostas Aire ka lauamängu tunni jaoks orienteeruva ajakava (vt lisa 7).

Lisaks reeglite teadmisele võiks lauamänguga ka lähemalt tutvuda, näiteks seda läbi mängida. Nii on võimalik väga hästi teada saada, millised küsimused võivad tekkida ning samuti saab lauamängu läbiviija ligikaudse hinnangu, kaua mäng ajaliselt kestab.

2.1.4. Tunni lõpetamine

Tunni lõpetamise osale tasuks samuti mõelda. Selle asemel, et lasta õpilastel tunnikella helisedes lahkuda, võiks pigem küsida tagasisidet läbiviidud tunni kohta või rääkida millestki, mis tunni sisu lühidalt kokku võtaks.

Juhul, kui soovitakse ka õpilastelt tagasisidet küsida, võiks koostada tagaside küsimustiku. Tagasiside küsimustik võib olla nii veebis kui ka paberil. Tagasiside küsimise korral on soovituslik tunni sisuline osa veidi varem lõpetada, et õpilastel oleks võimalus kohe tunnis tagasiside anda. Veebis olevat küsimustikku oleks õpilastel võimalus ka hiljem täita, kuid tõenäoliselt ei jää neil vastav aadress meelde või ei pea nad küsimustiku täitmist oluliseks ning unustavad seda hiljem teha. Näiteks koostasid 2013/2014. õppeaastal koolikülastuste läbiviijad veebiküsitluse, mille täitmiseks jäeti tunni lõpus aega.

2.2. Tunni läbiviimine

Tunni läbiviimise osas tuuakse täpsemalt välja peatükis 1.2. mainitud tunni etappides tehtavad tegevused. Samuti on välja toodud ka 2013/2014. õppeaastal läbiviidud koolikülastuste tundide üldine sisu.

Enne tunni alustamist tuleks välja otsida esitatavad slaidid. Kui tegemist on Pythoni töötoaga, siis võiks Pythoni IDLE's valida suurema tekstisuuruse, et ka õpilased teksti näeksid. Lauamängu töötoa jaoks võiks panna valmis mängulauad ja nupud.

2.2.1. Pythoni tunni läbiviimine

Sissejuhatus

Sissejuhatuses võiksid tunni läbiviijad ennast nimeliselt tutvustada ning mainida, kuidas nad on seotud infotehnoloogiaga. Seejärel võiks öelda õpilastele, miks üldse tuldi neid külastama ja tutvustada ka tunni ülesehitust. Kuna enamus õpilasi tõenäoliselt ei tea, mis on infotehnoloogia ja programmeerimine, siis võiks rääkida ka natukene üldisemalt

arvutiteadusest ning kirjeldada, mida kujutab endast programm. Seejärel saaks juba öelda, mis on programmeerimine.

Kui tegemist on gümnaasiumi klassiga, siis võiks õpilastele tuua välja ka Tartu Ülikooli informaatika erialale sisseastumiseks vajalikud eksamid ja nende osatähtsuse. Lisaks võib välja tuua ka õppekavas olevad ained, rääkida enda kogemusest ja mainida näiteks võimalikke töökohti. Loomulikult võib kajastada ka muud informaatikaga või ülikooliga seonduvat. Arvestada tuleks, et sissejuhatus ei oleks väga pikk, kuna ka programmeerimine võtab oma aja. Enne Pythoni töötoa programmeerimise osa juurde minekut, tooksin välja ka näidis-sissejuhatuse, mida kasutati 2013/2014. õppeaastal koolikülastuste tegijate poolt. Järgnev sissejuhatus on koostatud Tauno Paltsi poolt, kusjuures sissejuhatuse tegemisel on tuginetud Jaak Vilo videoloengule „Biti jõud on suur“ [3]. Sissejuhatuse slaide antud materjalis välja ei tooda, kui on soov neid näha, siis võiks ühendust võtta Tauno Paltsiga.

2013/2014. õppeaastal koolikülastuste tegijate sissejuhatus algas kõigepealt inimeste tutvustamisega. Seejärel toodi lühidalt välja sissejuhatuse osad, milleks olid:

1. Sissejuhatus arvutiteadusesse
2. Programmeerimine ja algoritmid
3. Tartu Ülikool ja üldine info

Esimeses sissejuhatuse osas selgitati õpilastele mõistet bitt, mis on kõige väiksem arvuti mälu ühik. Biti väärtus saab olla kas 0 või 1. Antud sissejuhatuses tähistati bitiga 0 sõna „ei“ ja bitiga 1 sõna „jah“. Seejärel küsiti õpilastelt küsimusi, millele paluti vastata biti väärtusega. Järgmisena toodi õpilastele välja, et alati ei taheta vastata küsimusele „ei“ või „jah“. Selle jaoks, et küsimusele saaks vastata „pigem ei“ ning „pigem jah“ võeti järgmiseks kasutusele 2 bitti. Õpilastel oli nüüd neli erinevat vastusevarianti, et küsimusele vastata. Siiski jäi ka neljast vastusevariandist väheks, kui õpilaselt küsiti, mis on tema nimi. Seega võeti kasutusele informatsiooniühik bait. Bait koosneb kaheksast bitist ning ühe baidi kirja panekuks on 2^8 ehk 256 võimalust. Näitena toodi välja, et baitidega saab ära defineerida kõik QWERTY klaviatuuril olevad sümbolid, numbrid ning suured ja väikesed tähed [4].

Edasi räägiti sissejuhatuses RGB (R-*red*, G-*green*, B-*blue*) värvimudelist, mis on kasutusel arvutites. Muutes kolme värvi esinemise intensiivsust, saab moodustada kõik teised värvid. Iga värv kasutab ühte baiti, järelikult kulub ühe tooni edastamiseks 3 baiti infot. Teades

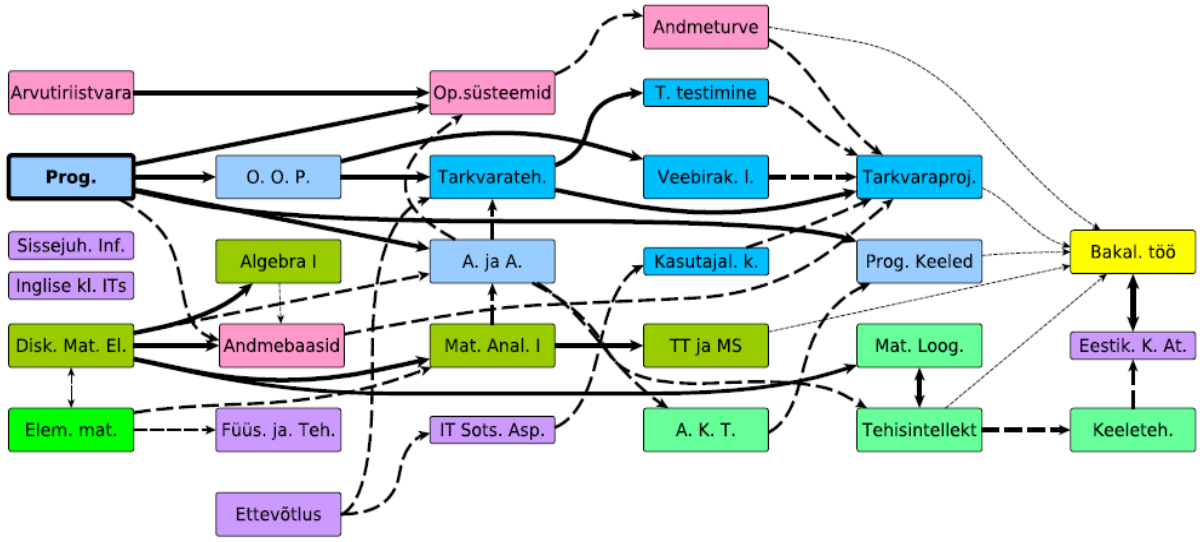
eelnevat infot, saab arvutada ligikaudse pildifaili suuruse, juhul kui teame pildifaili mõõtmeid. Näiteks, kui pildifaili suurus on 3600 x 2400 pikslit ehk 8,6 megapikslit ning me teame, et üks piksel on üks toon, mille edastamiseks kulub 3 baiti infot, saame pildifaili suuruseks ligikaudu 26 megabaiti [4].

Sissejuhatuse teises osas räägiti kõigepealt, mis on programm ja toodi ka lühike koodinäide. Õpilastele seletati, et kuigi arvuti loeb infot nullide ja ühtedena, siis tänapäeval programme enam nii ei kirjutata. Välja on töötatud erinevad programmeerimiskeeled, mille abil saab arvutile käsklusi anda. Lihtsustatud kujul öeldes tähendab programmeerimine, et arvuti täidab rida realt programmeerija poolt kirjutatud käske. Kindla ülesandega koodi osa nimetatakse algoritmiks. Näiteks parooli äratundmise algoritm. Sisestades õige parooli, lastakse kasutaja vastavasse keskkonda, vale parooli korral aga mitte. Arvutile antakse sisend klaviatuuri abil ning väljund ilmub ekraanile [4].

Õppimisvõimaluste tutvustamine (gümnaasiumile)

Gümnaasiumiklassidele tutvustati sissejuhatuse lõpus ka mõningaid infotehnoloogia sektori ametikohti. Samuti mainiti ära Tartu Ülikooli informaatika eriala sisseastumiseks vajalikud eksamid, milleks on eesti keele või eesti keel teise keelena riigieksam (osatähtsus on 40%) ning matemaatika lai riigieksam (osatähtsus on 60%). Sissejuhatuse lõpetuseks näidati õpilastele ka õppekava (joonis 1) ning räägiti lühidalt mõningatest läbitavatest ainetest.

Joonis 1 – Tartu Ülikooli informaatika eriala õppeained [4].



Algselt koostatud Margus Niitsoo poolt.

Joonisel (joonis 1) on toodud tulpade kaupa õpilastele välja ühes semestri õpitavad ained. Samuti on näha ainetevahelisi seoseid. Aine nimetusest väljuv tume nool näitab, et vastav aine on eeldusaineks noole teises otsas olevale ainele. Katkendlik joon näitab ainete seotust, mida tugevam on joon, seda rohkem on ained omavahel seotud [4].

Programmeerimiskeskkonna tutvustus ja näidisülesanne

Programmeerimiskeele Python tutvustamisel võiks mainida, et antud keel on esimene, millega puutub õpilane kokku Tartu Ülikooli informaatika erialal. Tunnis viibijatele võib rääkida ka üldisemalt, mille poolest erineb Python mõnest teisest programmeerimiskeelest, näiteks Javast. Üldisemalt on Pythonist kirjutatud peatükis 3.2.

Täpsemalt võiks rääkida, mida on plaanis õpilastega tunnis teha, milliseid Pythoni vahendeid kasutatakse ning miks just neid. Enne koodi kirjutamise alustamist, võiks ära mainida ka, kuidas Pythonit üldse avada ning millise nimega faili salvestada. Samuti võiks välja tuua, miks ei kirjutata tavaliselt koodi Pythoni graafikalisse kasutajaliidesesse (Python IDLE).

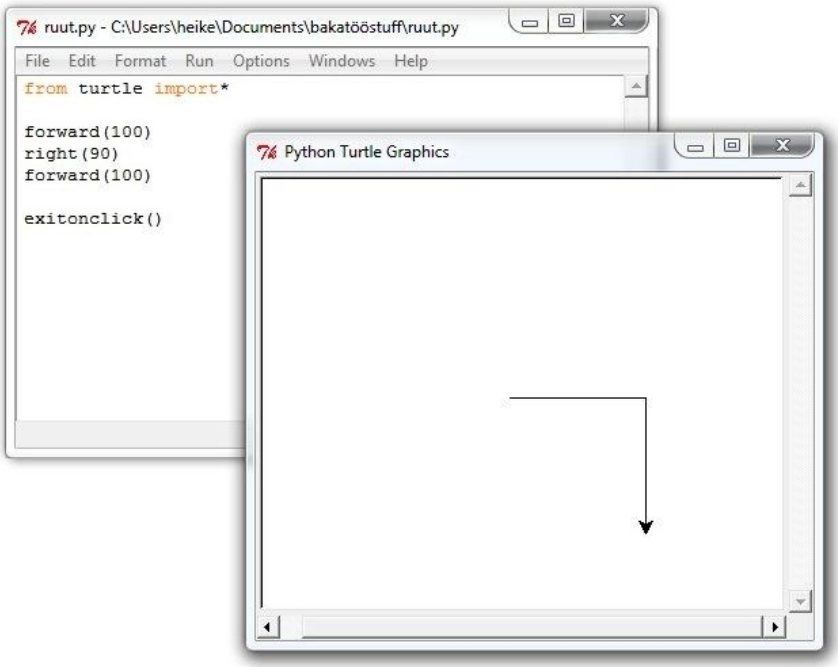
Programmeerimiskeele Python tutvustamist alustati läbiviidud koolikülastustes sellega, kuidas Pythonit üldse avada. Kõige kasutatavamaks mooduseks oli lihtsalt arvuti startmenüü otsingusse märgusõna „IDLE“ kirjutamine, mis avas IDLE’i käsurea. Seejärel prooviti IDLE’i

käsureal lihtsat arvutustehet (näiteks 3+4), et õpilased näeksid, kas vastus väljastatakse. Edasi seletati õpilastele, et tavaliselt siiski koodi käsureale ei kirjutata, kuna iga kord, kui soovitakse kontrollida, kas mingisugune osa koodist töötab, tuleb kogu kood uuesti kirjutada. Näiteks kui me tegelikult ei tahtnud koostada tehet 3+4, vaid hoopis 3+45, siis IDLE'i käsureal ei ole võimalik eelnevalt kirjutatud rida enam muuta, vaid tuleb sisestada uus rida.

Pärast IDLE'i käsurea katsetamist, näidati õpilastele, kuidas luua uut faili, kuhu nad saaksid oma koodi kirjutada. Mainiti ära, et failinime lõpus peab olema laiend „.py", et Python oskaks vastavat faili avada. Kuna Pythonis on olemas kilpkonna graafika (ingl. *turtle*), siis öeldi õpilastele ka, et nende poolt loodud faili nimeks ei tohiks panna „*turtle.py*", sest vastasel juhul satub Python faili avamisel segadusse.

Järgnevalt kirjutati õpilastele näidiseks mõned read koodi, mis sisaldasid kilpkonnagraafika mooduli importimist ning ruudu kahe külje joonistamise jaoks vajalikke käske (joonis 2). Tunnis läbiviidud ülesandeid vastavasse peatükki kirja ei pane, aga need on nähtaval õpilastele jagatud töölehel (vt lisa 1).

Joonis 2. Ruudu kaks külge kilpkonnagraafikas



autor: Heike Randlahe

Ülesannete lahendamine

Õpilastele võiks anda aega iseseisvalt ülesannete lahendamiseks. Selle aja jooksul on soovituslik töötoa läbiviijal klassis ringi käia ning vaadata, kaugel ülesannetega ollakse. Kui õpilasel esineb küsimusi või on näha, et ta on ühe ülesande juures väga kaua, võiks talle abi pakkuda. Võib tekkida ka olukordi, kus ainult rääkimisest ei piisa. Sellisel juhul võiks proovida võimaluse korral seletada joonistamise abil.

Võib leiduda õpilasi, kes ei soovi kaasa teha, kuid neid esineb vähe (2013/2014. õppeaastal korraldatud koolikülastusel oli neid kokku kaks või kolm). Programmeerimine ja üldse infotehnoloogia ei sobi kõigile. Sellisel juhul võiks vastavatel õpilastel lihtsalt lasta tegeleda neile huvitavate asjadega kokkuleppel, et nad ei hakka teisi õpilasi segama. Samuti võib esineda õpilasi, kes on varem Pythoniga kokku puutunud või kes saavad lihtsalt antud ülesannetega varem valmis. Sellistele õpilastele võiks anda lisaülesandeid.

Pythoni töötoa läbiviija võiks meeles pidada, et tunni eesmärgiks ei ole kõikide töölehel olevate ülesannete lahendamine. Tunni eesmärgiks on siiski õpilaste silmaringi laiendamine infotehnoloogia vallas ning programmeerimiskeele Python tutvustamine. Seega ei tasuks õpilasi ülesannete lahendamisel tagant kiirustada.

Läbiviidud koolikülastuste Pythoni töötoas lahendasid õpilased ülesandeid valdavalt töölehe (vt lisa 1) järgi. Nii mõneski koolis oli õpilasi, kes olid varem Pythoniga kokku puutunud ning töötoa läbiviijate poolt antud ülesanded tundusid neile lihtsad. Sageli nad tegid siiski kaasa, proovides joonistada erinevaid kujundeid tsüklite kasutamisega. Samuti anti vahel ka varem Pythoniga kokkupuutunud õpilastele teistsuguseid ja natukene keerulisemaid ülesandeid. Vastavaid ülesandeid antud bakalaureusetöös välja ei tooda. Huvi korral võib ühendust võtta 2013/2014. õppeaastal toimunud koolikülastuste läbiviijatega.

Tunni lõpetamine

Tunni lõppemise lähenemisest võiks õpilastele enne tunnikella helisemist teada anda. See annab õpilastele võimaluse veel pooleliolevaid ülesandeid lõpetada. Olenevalt sellest, kas palutakse õpilastel ka tagasisidet täita, tuleks valida, mitu minutit varem programmeerimise osa ära lõpetada. Lisaks tagasiside küsimisele võiks mainida ka ProgeTiigri lehekülge [2], kus on olemas Pythoni materjalid. Juhul, kui töötoa läbiviija ei soovi tagasisidet küsida, võiks teha

ikkagi tunni kohta kokkuvõtte. Näiteks uurida õpilastelt, mis oli nende jaoks kõige raskem ja mis neile kõige rohkem meeldis.

Antud bakalaureusetöös nimetatud koolikülastajad lõpetasid oma Pythoni tunni õpilastelt tagasiside küsimisega. Selle jaoks kasutasid nad veebis olevat tagasiside vormi, milles lisaks õpilaste soole ja vanusele küsiti ka järgnevat:

1. Millist programmeerimiskeelt tutvustati? (vastav küsimus oli tagasisides, kuna küsitlus oli ühine mõlema programmeerimiskeele töötoa jaoks)
2. Arvamus tutvustatud programmeerimiskeele kohta:
    a. oli sobiva tasemega
    b. oli liiga lihtne, oleks tahtnud mõne teise programmeerimiskeele tutvustust
    c. oli liiga keeruline, väga raske oli aru saada
3. Mis meeldis tunni läbiviimisel?
4. Mis ei meeldinud tunni läbiviimisel?
5. Kas tunnis osalemine muutis soovi programmeerimisega tegeleda?
    a. olin enne tundi sellest huvitatud ja tund suurendas seda huvi
    b. olin enne tundi sellest huvitatud, aga tund seda huvi ei suurendanud
    c. enne tundi huvi puudus, aga tund tekitas huvi
    d. enne tundi huvi puudus ja tund ka seda ei tekitanud

Tagaside täitmine oli anonüümne nii õpilaste kui ka kooli suhtes. Õpilaste poolt antud tagasiside vastuseid analüüsiti ning need on toodud välja 4. peatükis.

2.2.2. Scratchi tunni läbiviimine

Sissejuhatus

Scratchi tund on mõeldud pigem põhikooli II astmele. Sissejuhatus võib olla sama, mis Pythoni töötoa alguses, aga välja võiks jätta ülikooli sisseastumistingimustest ja õppekavast rääkimise. Lühidalt võiks ka seletada, mis on programmeerimine.

Näiteks rääkis Aire Kuresson enda poolt läbiviidud Scratchi töötoa alguses lühidalt sellest, mis läbiviidavas tunnis plaanis on. Seejärel andis ta võimaluse õpilastel öelda, mis on nende arvates programmeerimine. Kui keegi midagi pakkuda ei osanud, ütles Aire ise, et

programmeerimine on arvutile samm-sammult ütlemine, mida arvuti tegema peab. Edasi mindi Scratchi programmeerimise osa juurde.

Programmeerimiskeskkonna tutvustamine ja näidisülesanne

Scratchi programmeerimist saab teha veebis [5] või soovi korral võib Scratchi enda arvutisse tõmmata. Juhul kui kasutatakse veebis olevat võimalust, võiks esimese asjana näidata ära, kust on võimalik vahetada keelt. Oleks soovituslik, et kõikidel õpilastel on sama keelevalik, sest sellisel juhul on neil lihtsam järgida, millest räägitakse. Keel võiks olla valitud olenevalt koolist, näiteks venekeelses koolis võiks lasta õpilastel valida vene keele, kuigi see võib raskendada töötoa läbiviijal õpilaste aitamist.

Samuti võiks mainida, et soovi korral on õpilasel võimalik Scratchi veebilehel endale konto luua ning sellisel juhul saavad nad oma tehtud töid sinna salvestada ning neid ka hiljem avada. Enne programmeerimise juurde minekut võiks õpilastele näidata ka, kus miski asub.

Antud osas ei kirjutata täpsemalt, kuidas tutvustasid programmeerimiskeskkonda 2013/2014. õppeaastal Scratchi töötoa läbiviijad. Siiski on töö lisas (vt lisa 8) välja toodud, millistele programmeerimiskeele Scratch elementidele võiks tähelepanu pöörata.

Ülesannete lahendamine

Scratchi ülesannete lahendamisel võiks õpilastele jätta päris palju vabadust. Siiski võiksid olla mingid kindlad ülesanded, mida õpilastel tuleks teha. Kuna Scratchis on päris palju erinevaid võimalusi ning õpilastel läheb päris kaua aega, et nendega tutvuda, ei peaks andma õpilastele väga palju ülesandeid lahendamiseks.

Olenevalt klassi vanusest võib juhtuda ka, et osad tunnis olevad õpilased leiavad, et nende jaoks on Scratch liiga lihtne. Sellisel juhul võiks proovida anda neile mõningaid keerulisemaid ülesandeid kui teistele. Juhul, kui on näha, et neid tõesti programmeerimine ei huvita, ei tasuks neid siiski sundida kaasa tegema, kuna programmeerimine ei sobi kõigile.

Ülesandeid võiks õpilastele anda järk-järgult, kuna tõenäoliselt tekib ülesannete lahendamisel küsimusi ning selle asemel, et kõigile eraldi seletada (õpilased lahendavad üleandeid erineva kiirusega), saaks neid seletada klassi ees. Antud bakalaureusetöös mainitud koolikülastuste tegijad valmistasid ette kaks ülesannet. Alguses näidati ette, kuidas vastavat ülesannet

lahendada ning siis lasti õpilastel endil teha. Mingi aja möödudes näidati ette teine ülesanne. Kuigi õpilastele töölehti ei jagatud, olid need siiski tunni läbiviijal endal olemas (vt lisa 2).

Tunni lõpetamine

Scratchi tunni lõpetamisel võiks keskenduda samadele asjadele nagu Pythoni tunni lõpetamisel (täpsemalt punktis 2.2.1. Pythoni tunni läbiviimine, osas „tunni lõpetamine“). Scratchi tunni lõpus võiks lisaks kokkuvõtte tegemisele ja tagasiside küsimisele näidata ka võimalust, kuidas õpilased saaksid oma tööd salvestada ning hiljem arvutist tööd üles laadida. Samuti võiks vajadusel aidata õpilastel nende tööd neile endale meilile saata.

Scratchi töötoa õpilaste poolt antud tagasisidet on samuti analüüsitud ning see on toodud välja peatükis 4.

2.2.3. Lauamängu tunni läbiviimine

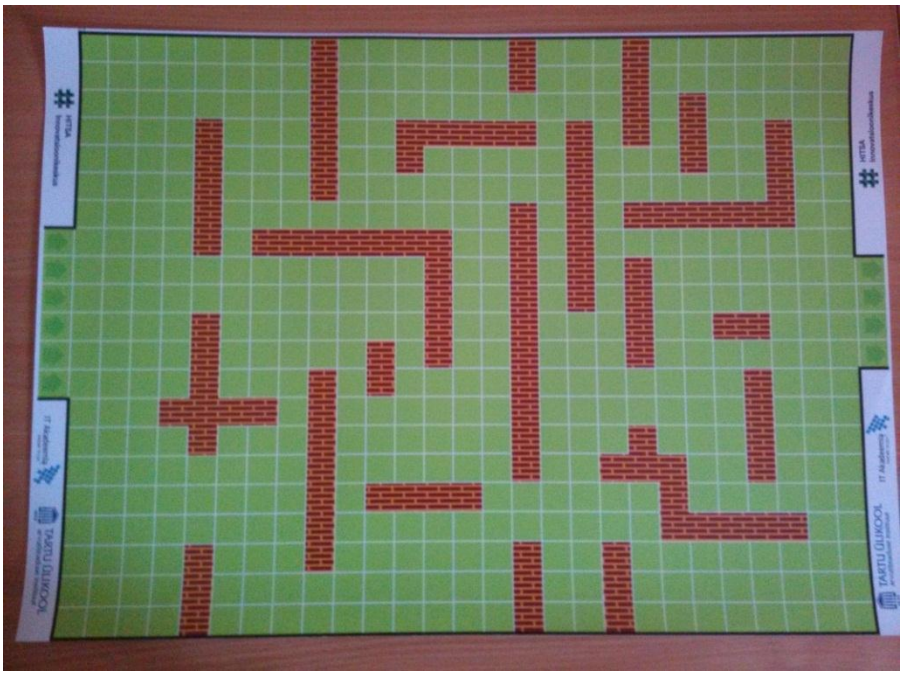
Sissejuhatus

Lauamängu töötoas mängiti mängu nimega „*The Robots Game*“ [2]. Lauamäng on seotud programmeerimisega, kuna iga mängija paneb enda käigud kirja selleks ette nähtud lehele (joonis 5). Kui mängija oma käigu ajal avastab, et tal ei ole võimalik käiku sooritada, siis ei ole tal võimalik ette antud käike enam muuta. Ka programmeerimisel ei ole võimalik enam käsklusi muuta, kui neid jooksutatakse. Juhul, kui mängija tahtis oma nuppu liigutada edasi 4 sammu, aga kirja pani kolm, siis teebki nupp kolm sammu edasi. Samamoodi on ka programmeerimisel, arvuti teeb need käsud, mis on kirjas, mitte neid käske, mida küll mõeldi, aga kirja ei pandud.

Mäng on mõeldud kuuele inimesele. Iga grupi jaoks on vaja:

1. Lauamängu laud (joonis 3).

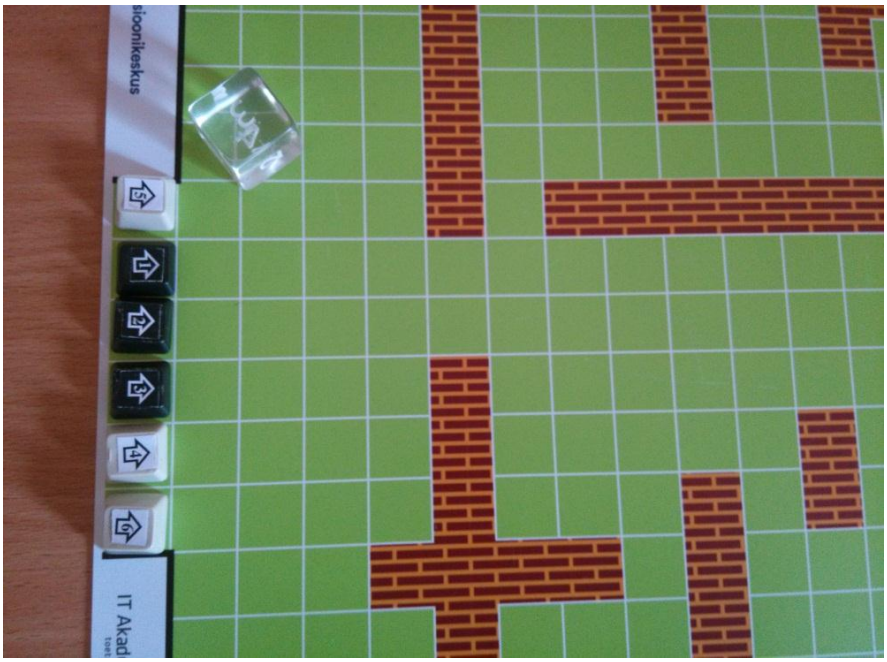
   Joonis 3 – „*The Robots Game*“ lauamängu laud



autor: Heike Randlahe

2. Kuus mängunuppu, mille peal on numbrid ühest kuueni, ning täring (joonis 4)

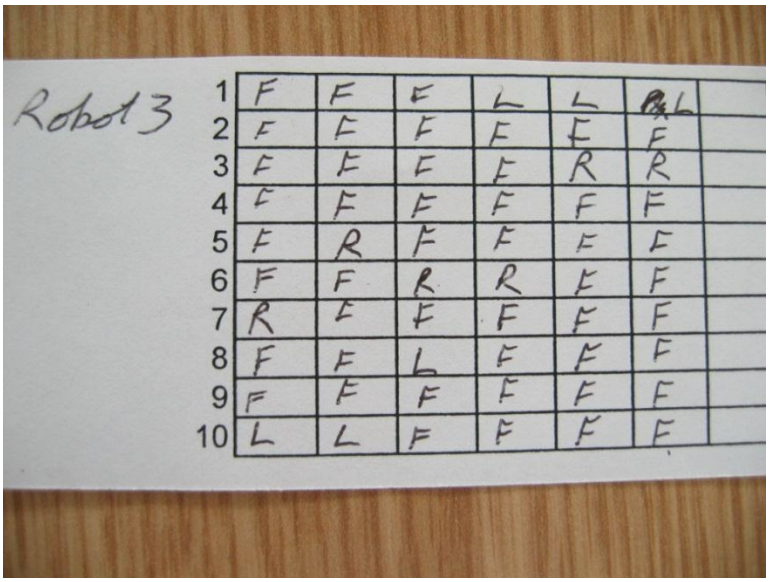
   Joonis 4 – Lauamängu nupud ja täring



autor:Heike Randlahe

3. Käikude lehed iga mängija jaoks (joonis 5)

Joonis 5 – Käikude lehe näidis [1]



Sissejuhatuses võiks jagada inimesed võistkondadesse ning seletada ära reeglid (vt lisa 6). Juhul, kui õpilaste arv ei jagu täpselt kuuega, võiks juhendaja ka ise mänguga liituda või paluda mõningatel õpilastel kahekesi ühe nupuga mängida. Seejärel võiks tuua välja ka mõningad mängu näidisolukorrad, et kontrollida, kas õpilased said reeglitest õigesti aru. Edasi võikski mänguga alustada.

Mängu mängimine

Mängu mängimise ajal võiks töötoa läbiviija kogu aeg jälgida, kas mängijad mängivad reeglite kohaselt. Vajadusel tuleks vastata õpilaste küsimustele ning lahendada vaidlusolukordi.

Tunni lõpetamine

Tunni lõpus võiks aega jätta ka kokkuvõtte tegemiseks. Kuna lauamängu viiakse tavaliselt läbi tavalises klassis mitte arvutiklassis, siis ei saa lasta õpilastel kohapeal tagasisidet täita. Siiski võib õpilaste arvamust tunni kohta küsida ka suuliselt.

Õpilastel ei pea laskma mängu lõpuni mängida. Võib olla võikski pigem mängu katkestada, kui see veel pooleli on, et õpilastel jääks võistlustahe. Tunni lõpuni jäävat aega võiks sisustada näiteks õpilastelt programmeerimisega seotud küsimuste küsimistega (vt lisa 8). Kindlasti võiks mainida, et üks lauamängukomplekt jääb ka koolile.

3. Programmeerimiskeel Scratch ja programmeerimiskeele Python kilpkonnagraafika

Selles peatükis räägitakse lähemalt programmeerimiskeelest Scratch ning Pythoni kilpkonna moodulist. Vastavates teemadest räägime lähemalt selleks, et lugejale oleks arusaadav, miks tutvustati just vastavaid keeli programmeerimisalaste koolikülastuste töötubades.

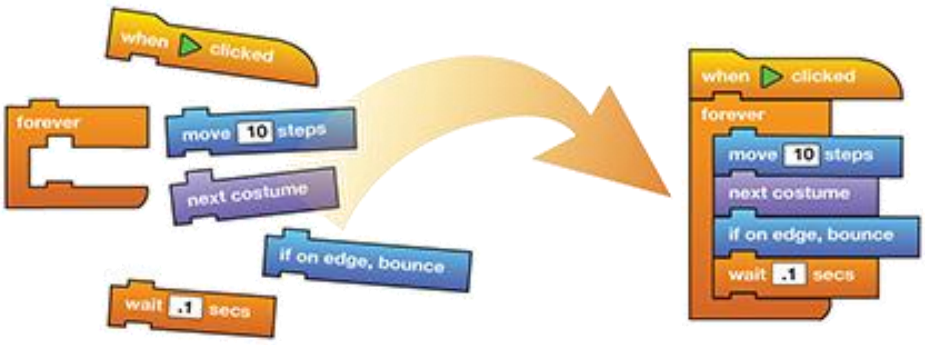
3.1. Scratch

3.1.1. Scratchist üldiselt

Esimene Scratchi versioon ilmus aastal 2003. Scratchi loomise põhieesmärgiks oli luua lähenemisviis programmeerimisele, mis tunduks huvitav ka inimestele, kes ei ole ennast kunagi varem programmeerijana ette kujutanud. Scratchi loomisel keskenduti sellele, et programmeerimiskeel oleks lihtne kõigile inimestele ning et kõigil oleks võimalus luua interaktiivseid programme (mäng, animatsioon jms) ning võimalus neid ka teistega jagada [7].

Scratchi loomisel võeti eeskujuks LEGO. Tehti programmeerimisplokid (ingl. *programming blocks*), mida kasutajad saaksid omavahel sobitada. Scratchi plokkidel on ühenduspesad, mis näitavad, millised plokid kuhu sobivad (vt joonis 5). See annab kasutajatele võimaluse ühendada omavahel erinevaid plokke ning näha, mis juhtub. Plokid on kujundatud selliselt, et nende kokkupanekul säiliks siiski koodi süntaktiline mõte.

Joonis 5. Scratchi plokkide omavahel sobitamine [8].



http://tinypic.com/view.php?pic=14marsn&s=5#.U2_ACfmSyN0

Nimetus „Scratch“ tuleb ingliskeelsest tegusõnast „*to scratch*“. See tuleb kriipimistehnikast, mida kasutavad DJ-d oma muusika kokku panemisel. Scratchi programmeerimisel kasutatakse sarnast meetodit – kokku on võimalik panna graafika, animatsioon, pildid, muusika ja erinevad helid [7].

Scratchis pannakse plokke kokku selleks ettenähtud alal. Koodi väljundit on näha laval. Samal ajal, kui koodi jooksutatakse, saab programmeerija teha muudatusi ning seega on lihtne katsetada erinevaid programmi võimalusi. Juhul kui soovitakse mõningaid asju paralleelselt jooksutada, tuleb teha erinevaid plokkide komplekte, mis kõik täidetakse samal tingimusel [7].

3.1.2. Miks õpetada Scratchi?

Kogenematute jaoks tunduvad traditsioonilised programmeerimiskeeled segadust tekitavate ingliskeelsete sõnade ning semikoolonite ja nendevaheliste süntaksite seoste jadana. Sageli piisab ühest pilgust koodile, et heidutada õpilast programmeerimise algteadmiste õppimisest.[8]

Scratch aga võimaldab õpilasel programmeerida kõigest hiire abil, liigutades ja ühendades plokke. Peatükis 3.1.1 sai mainitud, et plokke saab ühendada ainult nii, et säilib koodi süntaktiline mõte. See annab programmeerijale võimaluse õppida programmeerimise põhialuseid ja loogilist mõtlemist ning alles hiljem üritada süntaksist täielikult aru saada [8].

Scratchi abiga õpib kasutaja mõtlema nii nagu programmeerija, tutvustades talle erinevaid programmeerimise võimalusi lihtsalt ja mängulisel moel. Samuti on Scratchi puhul hea see, et õpilastel on võimalik Scratchi kodulehel teha endale kasutaja ning suhelda teiste Scratchi loojatega. See annab õpilastele võimaluse teistelt kasutajatelt abi saada ning samas ka uurida teiste kasutajate poolt loodud programmide koodi ning seeläbi saada juurde teadmisi [8].

3.2. Pythoni kilpkonnagraafika

3.2.1. Pythonist üldiselt

Python on interpreteeritav, interaktiivne, objektorienteeritud programmeerimiskeel. See sisaldab mooduleid, erindeid, dünaamilist kirjutamist, kõrgel tasemel dünaamilisi andmetüüpe ja klasse. Pythonis on seotud võimsus ja lihtne süntaks [9].

Python loodi 1990. aastal Guido van Rossumi poolt. Python sai oma nime seitsmekümnendate aastate BBC komöödiasarjast „*Monty Python's Flying Circus*“. Looja vajas nime, mis oleks lühike, ainulaadne ja natukene salapärane. Kuna talle meeldis komöödiasari, siis arvaski ta, et nimi oleks sobiv [9]. Python, nagu paljud teisedki skriptikeeled, on tasuta kasutatav isegi kaubanduslikel eesmärkidel ning samas saab Pythonit kasutada peaaegu kõikidel arvutitel [10].

3.2.2. Miks õpetada Pythoni kilpkonnagraafikat?

Mis on kilpkonn (*turtle*)?

Programmeerimise tulemuse visualiseerimiseks kasutatakse vahel n-ö küberneetilist looma – sageli kilpkonna. Esmakordselt kasutati seda lastele mõeldud programmeerimiskeeles LOGO. Kilpkonna kasutamise eesmärgiks on lihtsustada programmeerimist ja mõtlemist. Pythoni kilpkonnagraafika on inspireeritud LOGO programmeerimiskeelest, kuid on sellest eraldiseisev Pythoni moodul [11].

Pythoni kilpkonna õpetamine

Esimene kilpkonna kasutamine algab sageli sellest, et näidatakse õpilastele, kuidas kilpkonna käskude abil liigutada. Näiteks käsk „*Forward* 100“ liigutab kilpkonna 100 kilpkonna sammu (iga samm on umbes 1 millimeeter) sirgel edasi. Kirjutades „*Right* 90“, keerab kilpkonn 90 kraadi paremale. Kindlasti läheb natuke aega enne, kui õpilane saab aru, mida tähendavad numbrid käskude taga. Juhul kui ülesannet piisavalt huvitavaks teha, on kilpkonn hea võimalus lastele programmeerimise õpetamiseks [11].

Lisaks eelpool mainitule, oli programmeerimiskeele Python tutvustamine hea valik, kuna hetkel on see esimene programmeerimiskeel, millega puutub õpilane kokku Tartu Ülikooli õppima tulles. Kilpkonn ei ole küll esimene asi, mida aine raames õpitakse, aga üsna alguses

on ta küll. Samuti annab kilpkonn hea arusaama sellest, kuidas arvuti kasutaja poolt kirja pandud käske täidab.

4. Vajalikud ressursid

Antud peatükis pööratakse tähelepanu, mida oleks vaja, et programmeerimisalased koolikülastused edukalt läbi viia. Keskendutakse külastuse jaoks vajalikele ressurssidele: külastatavas koolis olemasolevatele asjadele, koolikülastaja omadustele ja oskustele ning ajale.

4.1. Soovitud ressursid külastatava kooli poolt

Selles osas räägime täpsemalt, mis peaks olemas olema koolis, kuhu minnakse programmeerimisalast koolikülastust tegema.

Kui tegemist on programmeerimist tutvustava töötoaga, võiks olemas olla:

1. Arvutiklass. Kui tegemist on Pythoni töötoaga, võiks paluda koolil ka kasutatavatesse arvutisse installeerida versiooni, mida kasutab Pythoni töötoa läbiviija (sellest võiks kooli varem teavitada). Meie kasutasime Pythoni versiooni 2.7.3.
2. Dataprojektor. Kuna õpilastele tuleks tutvustada programmeerimiskeskkonda ja võib-olla teha ka näidisülesanne, siis võiks saada neid suurelt näidata.

Lauamängu töötoa jaoks oleks vaja:

1. Klassiruum, mis võimaldab laudu ümber tõsta, kuna tõenäoliselt ei mahu kuus õpilast mugavalt ühe laua taha ära.
2. Dataprojektor. Seda juhul, kui töötoa läbiviija soovib õpilastele midagi suurelt ekraanilt näidata, näiteks reegleid või lubatud käike.

4.2. Koolikülastaja omadused ja oskused

Vastavas osas tuuakse välja täpsemalt mõningad omadused ja oskused, mis võiksid olemas olla koolikülastusi tegeval inimesel. Väljatoodud oskused ja omadused on saadud intervjueerides 2013/2014. õppeaastal koolikülastuste läbiviijaid Aire Kuressoni ja Tauno Paltsi.

Kannatlikkus

Kõige olulisema omadusena toodi välja kannatlikkus. Seda sellepärast, et võib ette tulla olukordi, kus õpilastele tuleb ühte asja mitu korda seletada. Sellisel juhul peaks jääma

rahulikuks ja seletama nii kaua, kuni õpilane aru saab. Seega ei sobiks koolikülastust tegema inimene, kes ärritub, kui tal tuleb ühte ja sama asja mitu korda seletada.

Sõbralikkus

Sõbralikkus on oluline mitmel põhjusel. Esiteks sellepärast, et koolikülastaja on ikkagi võõras inimene ning see kuidas ta käitub, võib kujundada ka õpilaste esmamulje ülikoolist ja seal õppivatest inimestest. Tõenäoliselt ei jäta kurjad ja mitteabivalmid töötoa läbiviijad head muljet ülikoolist ning lisaks ei tekita ka huvi programmeerimise vastu. Sõbralik ja abivalmis inimene tekitab ka teistele parema tuju ning üldisema meeldiva õhkkonna. Õpilased on avatumad ning julgevad abi küsida.

Oskus asju seletada

Siinkohal võib lugeja mõelda, et mis saab ikka asjade seletamises rasket olla. Iseenesest ei olegi midagi rasket. Tuleb siiski meeles pidada, et räägitakse inimestega, kellest paljud ei pruugi olla programmeerimisega kokku puutunud ning kui kasutada valdkonnale omast sõnavara, võib juhtuda, et töötoa läbiviijast ei saada aru. Samuti sai eelpool kannatlikkuse punkti all välja toodud, et võib-olla tuleb õpilastele asju mitu korda seletada. Juhul, kui õpilane ei saa ikkagi aru, mida või kuidas täpselt teha tuleb, tuleks proovida talle seda teistmoodi seletada.

Kuuldav hääl

Töötoa läbiviija peaks suutma õpilastest üle rääkida ning panna nad ennast kuulama. Olukord, kus võib valju häält vaja minna, on näiteks lauamängu töötoas. Võib juhtuda, et lapsed on mängust väga hasardis ja vaidlevad või lihtsalt räägivad pidevalt. Kui lapsi on palju, siis tekib sellest juba üsna suur müra. Tunni lõpus oleks siiski soovituslik mingi kokkuvõte teha ja sellel ajal peaksid lapsed kuulama, ehk siis tuleks teha piisavalt kõva häält, et õpilased taas töötoa läbiviijale tähelepanu pööraks.

Tahtmine ja huvi

Loomulikult võiks programmeerimisalaste koolikülastuste tegijail olla tahtmine ja huvi neid koolikülastusi läbi viia. Isegi, kui kõik eelpool mainitud oskused ja omadused on olemas, siis

tõenäoliselt ei ole koolikülastus edukas, kui töötubade läbiviijail puudub üldse huvi ja tahtmine koolikülastusi teha.

4.3. Aeg

Aeg on tõenäoliselt kõige määravam ressurss programmeerimisalaste koolikülastuste juures, sest ta määrab ära, kas üldse ja kui palju saab koolikülastusi läbi viia. Vastavas peatükis keskendutakse sellele, kui kaua võtab aega koolikülastuse tegemine. Siinkohal on vaatluse alt välja jäänud koolikülastuste ettevalmistamisele kuluv aeg.

Meie koolikülastuste grupp külastas kümne päevaga kümmet kooli. Järgnevalt vaadeldakse, kaua läheb aega programmeerimisalaste koolikülastuste tegemiseks rohkemates koolides. Vaatluse alla võetakse kaks versiooni. Esimene neist keskendub sellele, et külastatakse kõiki üldhariduskoole. Teine versioon oleks bakalaureusetöö autori arvates reaalsem järgmiste koolikülastuste läbiviimiseks. Järgnevates arvutustes kasutusel olevad andmed on kõik pärit Eesti Hariduse Infosüsteemi leheküljelt [12], kui ei ole just öeldud teisiti.

4.3.1. Esimene versioon – külastame kõiki üldhariduskoole

Eestis on praeguse seisuga 559 üldhariduskooli [12]. Kui külastada koole samas tempos, nagu meie seda tegime, ehk siis kool päevas, kuluks programmeerimisalaste külastuste jaoks 559 päeva. Ühes õppeaastas on keskmiselt 175 koolipäeva [13]. Järelikult kuluks kõikide koolide külastamiseks 3,2 õppeaastat, eeldusel, et jõutakse külastada üks kool päevas ja ühtegi koolipäeva ei jäeta vahele. Koolide hulgas on 13 kooli, milles on ainult gümnaasiumiosa ning 220 kooli, milles on ainult põhikooliosa [12].

Programmeerimisalaste koolikülastuste tegemiseks kõikides koolides läheks tõesti 559 päeva, kui igas päevas külastada ühte kooli. Samas aga tuleb arvestada, et koolipäeva pikkuseks on maksimaalselt 8 koolitundi (koolitunni tavaline pikkus on 45 minutit). Kas tõesti jõutakse 8 tunniga külastada ühte kooli? Kindlasti jõutakse, kui eesmärgiks ei ole anda tunde kõigile. Vaatame aga lähemalt, kaua kuluks aega, et anda programmeerimisalane tund igale klassile alates II põhikooli astmest.

Tavaliselt on koolides üks arvutiklass, maksimaalselt kaks. Järgmiste arvutuste tegemiseks eeldame, et koolis on üks arvutiklass, kuhu mahub ära üks klassitäis õpilasi. Kõige pealt võtame vaatluse alla koolid, milles on ainult gümnaasiumiosa. Antud arvutuseks võtame, et

kõikides gümnaasiumites on viis paralleeli. Tegelikult on see arv muidugi kooliti erinev. Kui gümnaasiumis on viis paralleeli, siis kokku on ühes gümnaasiumis 15 klassi, mis aga tähendab seda, et koolis programmeerimisalase tunni andmiseks kulub 2 päeva juhul, kui tunni pikkuseks on 45 minutit. Paljudes gümnaasiumites on tunni pikkus suurem (näiteks 90 minutit). Seega läheks ühe kooli jaoks aega 4 päeva. Kõikide gümnaasiumite jaoks läheks 52 päeva.

Järgnevalt võtame vaatluse alla koolid, kus on ainult põhikooliosa. Eelnevalt sai välja toodud, et selliseid koole on 220. Kui tunde antakse alates II põhikooli astmest, siis on klasse kokku 9, kusjuures eeldame jällegi, et igal klassil on 3 paralleeli, seega ühes koolis on vaja anda tunde 18 klassile. Põhikoolis võtame ühe programmeerimisalase tunni pikkuseks 45 minutit, seega päevas saaksime anda 8 tundi. Järelikult ühe kooli külastamiseks kuluks meil 3 päeva ja 220 kooli külastamiseks kuluks 660 päeva, mis, nagu näha, on suurem vaatluse all olevate koolide arvust.

Siiamaani oleme saanud, et 13 gümnaasiumi ja 220 põhikooli külastamiseks, kuluks 712 päeva. Nüüd on meil jäänud arvutada ainult veel ülejäänud 326 kooli külastamiseks vajaminev aeg. Nende koolide jaoks võtame, et paralleele on 3 ja tunni pikkuseks nii põhikooli- kui ka gümnaasiumiosas on 45 minutit. Seega ühes koolis on vaja anda tunde 27 klassile ning kuna päevas saaksime anda maksimaalselt 8 tundi, siis kuluks meil ühe kooli jaoks 4 päeva ning 326 kooli jaoks kuluks 1304 päeva. Kõikide arvude kokku liitmisel saame, et 559 kooli külastamiseks, kuluks meil 2016 päeva. Võtame endiselt, et õppeaastas on 175 koolipäeva ning seega kuluks meil programmeerimisalaste koolikülastuste tegemiseks ligikaudu 11,5 õppeaastat.

Kogu selle arvutuse juures tasub muidugi arvesse võtta, et paralleelide arv koolides on tegelikult teine ning samuti pole arvesse võetud olukordi, kus arvutiklasse on rohkem või ühte klassi mahub näiteks kahe klassi võrra õpilasi või saab samal ajal anda tundi mitmes erinevas klassis või isegi mitmes erinevas koolis, mis oleneb täielikult koolide võimalustest ja loomulikult koolikülastuste tegijate arvust. Samuti on ka tundide pikkus oletuslik, kuna kooliti võivad need väga erineda ning lisaks on eeldatud, et õppeaasta igal päeval on võimalik kuhugi kooli külastust teha. Kogu see arvutus on tegelikult oletuslik ja on antud töös toodud ära lihtsalt selle jaoks, et näidata, kui kaua kulub aega, et teha programmeerimisalaseid

koolikülastusi kõikidesse koolidesse. Antud arvutuse juures ei ole ka arvesse võetud, et klassid vahetuvad iga aasta ja seega peaks koolikülastusi jäämagi tegema.

4.3.2. Teine versioon – reaalsem koolikülastuste läbiviimine

Antud programmeerimisalaste koolikülastuste korraldamisel on meie põhiliseks sihtgrupiks gümnaasiumid ning nendest just 11. klassid. Peame 11. klasse kõige sobivamaks sihtgrupiks, kuna nad on jõudnud juba koolieluga harjuda ning peavad hakkama mõtlema sellele, milliseid eksameid valida. Õppeaasta esimesel poolel oleks väga sobilik koolikülastuste jaoks 12. klass. Seda just sellepärast, et eksamivalik tuleb ära teha jaanuaris [14]. Hilisemad programmeerimisalased koolikülastused on kindlasti neile siiski kasulikud, kuid huvi äratamiseks on siis juba natukene hilja, sest eksamivalik on lõplik ning kui õpilane ei ole valinud matemaatika laia riigieksamit, siis kahjuks ei saa ta Tartu Ülikooli vastavale erialale astuda.

Loomulikult on programmeerimisalased koolikülastused kasulikud ka noorematele klassidele ning kindlasti ei jäta me neid arvestamata. Järgmistes arvutustes võtamegi vaatluse alla 13 gümnaasiumi ja 326 põhikool-gümnaasiumi. Programmeerimisalaseid koolikülastusi viime läbi endiselt alates 4. klassist.

Arutades reaalselt võimalike programmeerimisalaste koolikülastuste arvu õppeaastas Tauno Paltsiga, leidsime, et väga hea saavutus oleks, kui jõuaks õppeaastas külastada 30 kooli. Nendest 10 võiksid olla gümnaasiumid ja 20 põhikool-gümnaasiumid. Iga kooli jaoks arvestame siiski ühe koolipäeva ja maksimaalselt kuus 45-minutilist tundi. See tähendab muidugi seda, et programmeerimisalasest koolikülastusest ei saa osa võtta kõik klassid. Seega peaks jääma klasside valik külastatava kooli enda otsustada. Juhul, kui kool soovib nii programmeerimiskeelte Scratch ja Python tunde kui ka lauamängutunde, on võimalik külastajatel jagada teavet ja tekitada huvi rohkemates klassides. Seda muidugi eeldusel, et tunnid saavad toimuda paralleelselt.

Õppeaasta kestus on ligikaudu 9 kuud. Seega peaks ühe kuus tegema vähemalt 3 programmeerimisalast koolikülastust. Eeldusel, et üks kool võib soovida korraga nii kahe programmeerimiskeele kui ka lauamängu tundi, oleks ideaalne, kui koolikülastajad oleksid kuueliikmelistes meeskondades ning meeskondi võiks kokku olla kolm. Seda sellepärast, et esiteks peaks üks meeskond külastama kuu aja jooksul ainult ühte kooli, vahel ka kahte ning

teiseks sellepärast, et kuueliikmeline meeskond on praktiline. Igas meeskonnas võiks olla 2 Scratchi tundjat, 2 Pythoni tundjat ning 2 lauamängu läbiviijat. Seda just sellepärast, et juhul, kui keegi meeskonnaliikmetest ei saa kohale tulla, on vähemalt keegi, kes vastavat töötuba läbi saab viia ning juhul, kui korraga on kohal mõlemad töötoa läbiviijad, on neil lihtsam kõikide õpilasteni jõuda ja neid aidata.

5. Külastatud koolide poolne tagasiside

Antud peatükis on analüüsitud õpilaste poolt saadud tagasiside vastuseid ning samuti on toodud välja õpetajatepoolne tagasiside ja soovitused järgmistele koolikülastuste tegijatele.

5.1. Õpilaste tagasiside

Õpilastel lasti tagasiside täita programmeerimist tutvustava tunni lõpus. Tagasiside täitmine ei olnud kohustuslik ja seega on osade tunnis viibinute tagasiside puudu. Küsitlus asus *Google* veebikeskkonnas. Saadud küsimuste vastuseid analüüsiti ning järgnevalt tuuakse välja koolikülastuste tegijate meelest olulisim.

Koolikülastuste raames külastati kümmet Eesti kooli. Tabelis (tabel 1) tuuakse välja ligikaudne õpilaste arv igas koolis.

Tabel 1. Külastatud koolides õpilasi [15]

Valga Vene Gümnaasium	100
Sindi Gümnaasium	50
Tartu Kunstigümnaasium	80
Räpina Ühisgümnaasium	45
Vastseliina Gümnaasium	80
Jaan Poska Gümnaasium	60
Rõuge põhikool	85
Paide Ühisgümnaasium	120
Lähte Gümnaasium	55
Tartu Kommertsgümnaasium	25
KOKKU	**700**

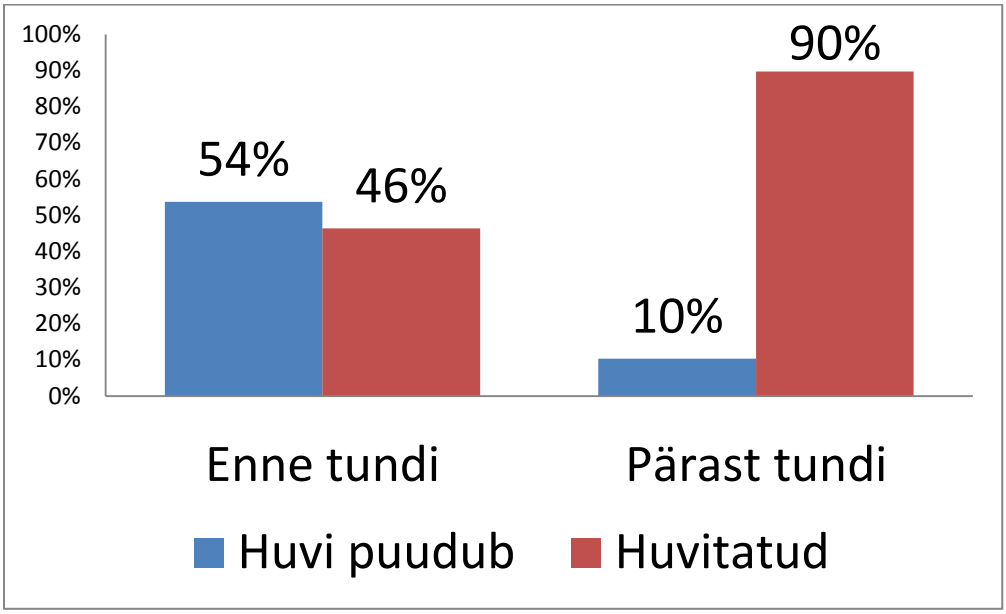
autor: Tauno Palts

Programmeerimisalastele koolikülastustele andis tagasisidet 159 õpilast [16]. Nendest tüdrukuid oli 38% ning poisse 62% [15]. Tagasisidet andud õpilaste hulgast 41% õpilastest viibisid Scratchi töötoas ning 59% õpilastest Pythoni töötoas [15]. Eelnevalt sai mainitud, et lauamängu töötoas õpilastelt tagasisidet ei küsitud ja seetõttu puuduvad andmed lauamängu töötoas olevatest õpilastest.

Tagasisides küsiti ka, kui keeruliseks õpilased vastava programmeerimiskeele töötuba pidasid. 87% vastanutest leidis, et nende jaoks oli läbiviidud töötuba sobiva raskustasemega. 9% õpilastest leidis, et programmeerimiskeel oli nende jaoks liiga lihtne ning 4% leidis, et vastav programmeerimiskeel oli liiga keeruline [15].

Samuti küsiti õpilastelt, kuidas oli nende huvi töötoas õpetatud programmeerimiskeele vastu enne tundi ning kuidas pärast tundi. Vastavad tulemused on kantud tabelisse (tabel 2).

Tabel 2. Õpilaste huvi programmeerimise vastu [15]



autor: Tauno Palts

5.2. Õpetajate tagasiside

Õpetajatelt küsiti tagasisidet mõned kuud pärast koolikülastuse läbiviimist. Seda sellepärast, et saada teada, kas midagi on ka läbiviidud koolikülastustes meeles. Tagasiside küsimused saadeti iga kooli õpetajale meili teel ning vastused saadi seitsmelt koolilt. Iga küsimuse juures on välja toodud mõned huvitavamad õpetajatelt saadud vastused. Täpsema tagasiside lugemiseks tuleks võtta ühendust bakalaureusetöö autoriga.

Kuidas otsustati kutsuda meid külastust tegema?

Selle küsimuse puhul tõid mõned õpetajad välja ProgeTiigri ja Tagasi kooli lehekülge. Ühte kooli kutsuti meid aga koolikülastust tegema valla volikogu esimehe soovitusel.

Miks just nendele klassidele?

Klasside valikul sai otsustavaks nii see, et õpilased saavad programmeerimise kursusi, kui ka see, et õpetaja õpetab ise vastavaid klasse. Mõnes koolis sai otsustavaks ka õpilaste enda huvi asja vastu.

Millised olid Teie ootused koolikülastuse suhtes?

Koolikülastustest oodati õpilaste huvi suurendamist programmeerimise vastu ja vaheldust tavalisest koolitunnist, mis oleks siiski hariva sisuga. Samuti loodeti, et tunde viiksid läbi inimesed, kes teavad, millest nad räägivad.

Milline oli õpilaste poolne tagasiside läbiviidud tunnile?

Õpilaste poolne tagasiside oli positiivne. Leidub õpilasi, kes jätkavad iseseisvalt programmeerimiskeelega tutvumist ja katsetamist ning toodi välja ka, et isegi tüdrukud leidsid programmeerimise enda jaoks huvitavana.

Kuidas jäite programmeerimist tutvustava külastusega rahule?

Õpetajad jäid külastusega rahule, kuna nägid ja kuulsid seda, mida olid lootnud. Samuti meeldis õpetajatele, et programmeerimisalaste koolikülastuste tegijad jõudsid igale õpilasele tähelepanu pöörata.

Mida positiivset jäi Teile silma tunni läbiviimise juures? Mida võiks edaspidigi samamoodi teha?

Positiivse tagasisidena toodi välja, et kõik koolikülastuste tegijad olid oma rollid ära jaganud ning jõudsid igale õpilasele tähelepanu pöörata. Lisaks meeldis tunni läbiviimise juures ka see, et tunnis mängitud lauamäng kingiti koolile ning kuna mäng on õpetlik, saab seda ka koolitunnis kasutada.

Mida võiks tunni läbiviimisele ette heita? Millele peaks rohkem tähelepanu pöörama?

Antud küsimuse vastusena toodi välja, et mõne ülesande jaoks võiks rohkem aega varuda. Samuti mainiti, et oleks tore, kui koolikülastused oleks pakutavad ka mõnes teises programmeerimiskeeles.

Mida uut saite Teie tunnis teada?

Õpetajad said tunnis targemaks Pythoni koha pealt. Lisaks saadi teada, millised on informaatika eriala sisseastumistingimused Tartu Ülikoolis.

Milline peaks olema programmeerimise koht koolide õppekavas? (Palun valige järgnevate valikute seast ning põhjendage oma vastust).

- **kõigile kohustuslik**
- **kohustuslik teatud õppesuundadele (millistele?)**
- **valitav**
- **üldse mitte**

Külastatud koolide õpetajad arvavad, et programmeerimine peaks olema kohustuslik reaal-, loodus-, ja tehnoloogiakallakuga suundadele. Lisaks võiks olla programmeerimine teistele suundadele ja põhikooliklassidele valikuline.

Milline on Teie arvates sobiv kooliaste, kus võiks alustada programmeerimise õpetamisega (kui üldse)?

Selle küsimuse vastustena toodi välja erinevaid variante, milles kokku nimetati kõiki kooliastmeid. Siiski leidsid kõik õpetajad, et programmeerimise õpetamisega võiks ikkagi enne ülikooli minekut alustada.

Millised on võimalikud takistused programmeerimise laialdasemal õpetamisel koolides?

Võimalike takistustena toodi välja nii juhtkonna kui ka õpetaja vähest huvi programmeerimise õpetamiseks. Samuti mainiti, et tunniplaani lihtsalt ei mahu, kuna muid aineid on niigi palju. Lisaks napib õpetajaid ja ka rahalised võimalused on napid. Takistusena toodi välja ka riistvaralisi mahajäämusi.

6. Muljed ja tähelepanekud

Selles peatükis on välja toodud programmeerimisalaste koolikülastuste läbiviijate Aire Kuressoni ja Tauno Paltsi muljed ja tähelepanekud. Mõlema koolikülastajaga viidi läbi intervjuu ning järgnevalt tuuaksegi välja need intervjuu küsimused ja vastused, mida pole eelnevates peatükkides kajastatud. Kuna tegemist on intervjuu küsimuste vastustega, siis on järgnev tekst mõnevõrra kõnekeelsem.

Kuidas sattusite programmeerimisalaseid koolikülastusi tegema?

Tauno sattus koolikülastusi tegema, kuna need on seotud tema tööga ning talle anti lihtsalt ülesandeks neid läbi viia. Koolikülastuste läbiviija Aire aga sattus vastavasse projekti aine informaatika didaktika raames. Nimetatud aine raames oli tal vaja anda mõni tund kas ülikoolis või koolis. Kuna Aire ei soovinud anda tunde ülikoolis ning sobivat kooli ka kahjuks ei leidnud ja juhtumisi oli algamas just programmeerimisalaste koolikülastuste projekt, siis pakuti talle võimalust sellest osa võtta.

Kuidas valmistusid ette koolikülastuseks?

Kuna Aire poolt tehtavad ettevalmistused said juba mainitud (vt. peatükk 3.1.), siis siinkohal toome välja Tauno Paltsi poolt sooritatud ettevalmistused. Enne koolikülastuste tegemist uuris Tauno, kuidas on sarnaseid koolikülastusi läbi viidud mujal maailmas. Samuti mõtles Tauno ka selle peale, kuidas oleks võimalik viia tundi läbi arvutiklassis ja tavalises klassis. Lisaks tundide sisulise poole peale mõtlemisele, kulus Taunol aega ka programmeerimisalaste koolikülastuste organiseerimisele. Tema oli see isik, kes suhtles koolidega ja korraldas koolikülastuste tegijate vahelised kokkusaamised. Samuti tegeles ta lauamängude printimisega ning õpilastele jagatavate meenete muretsemisega.

Mida mäletad esimesest koolikülastusest?

Esimene programmeerimisalane koolikülastus toimus detsembri lõpus. Intervjuu läbiviimise ajaks oli sellest juba mitu kuud möödas ning seetõttu uuriti, mida mäletavad Aire ja Tauno esimesest koolikülastusest. Kasutusel olevad uued materjalid tekitasid Taunos küll natukene segadust, kuid nähes sära ja huvitatust õpilaste silmis, teadis ta, et teeb siiski õiget asja. Lisaks sai esimese koolikülastuse käigus selgeks, millele võib järgmine kord vähem tähelepanu pöörata ja mida tasuks võib-olla natukene põhjalikumalt õpetada.

Airele meenus esimese asjana õpilaste reaktsioon. Tegemist oli neljanda ja viienda klassi poiste tunniga, mida viis Aire läbi koos bakalaureusetöö autoriga. Klassi astudes oli õpilaste esimene reaktsioon, et poisid pidid ju tulema. Sellist reaktsiooni Aire küll ei osanud oodata. Tund sujus siiski hästi ja oli näha, et õpilased olid asjast huvitatud, sest nad hakkasid kohe ise hästi aktiivselt ülesandeid lahendama.

Kuidas erineb programmeerimist tutvustav tund tavalisest koolitunnist?

Aire ütles, et ta ei ole andnud ega näinud tavalist informaatikatundi ning seega ei oska ta niimoodi võrrelda. Siiski arvab Aire, et erinevus on näiteks selles, et koolikülastuse tunnis ei ole õpilasel kohustust kaasa teha, aga tavalises tunnis on konkreetsemad eesmärgid, kas saada valmis mingi ülesanne või jõuda mingi kindla tulemuseni. Tauno lisas omalt poolt, et võrreldes tavalise koolitunniga on antav tund intensiivsem, kuna ühe programmeerimisalase koolitunni jooksul tuleb anda ülevaade mingisugusest teemast.

Kuidas mõjutas õpilaste vanus tunni andmist?

Tauno Palts väitis, et vanemad õpilased said ülesannetega kiiremini valmis ja neile tuli vahel anda lisaülesandeid. Airele jäi mulje, et nooremad õpilased olid tundidest rohkem huvitatud kui vanemad õpilased. Samuti tõi ta välja, et õpilase huvi tunnis olemise vastu sõltus ka sellest, kuidas oli kool valinud, millistele õpilastele tundi antakse. Osades koolides oldi otsustatud, mis klassid tunnis on ning teistes koolides said õpilased tundi registreerida. Aire arvas ka, et mõnes mõttes oli määravaks ka, kas õpilane oli varem programmeerimisega kokku puutunud. Näiteks neil õpilastel, kellel huvi ei olnud, oli ka tunnis pisut keerulisem olla.

Palun too näiteid erinevatest õpilastetüüpidest.

Nii Aire kui Tauno tõid välja, et õpilasi saab mitut moodi liigitada. Näiteks saab liigitada tunnist huvitatuse järgi. Leidub õpilasi, kes ei taha midagi teha ja siis on selliseid, kes ei ole eelnevalt teemaga kokku puutunud, aga tunni jooksul tekib neil huvi ning siis on veel õpilased, kes olid juba varem asjast huvitatud ja teevad huviga kaasa. Aire tõi välja selle, et ka abi küsimise poole pealt vaadatud on õpilased erinevad. On neid, kes annavad ise märku, kui neil tekib mingi küsimus või probleem. Teised on jällegi sellised, et nad üle klassi abi ei küsi, aga kui nende juurde minna ning küsida, kas nad vajavad abi, siis ütlevad, et oleks vaja küll.

Lisaks on veel õpilasi, kelle puhul on näha, et nad on kaua ühe ülesandega tegelenud, aga kui küsida, kas nad vajavad abi, siis nad ütlevad, et ei vaja ja küsivad pigem naabri käest.

Millises vanuses õpilasi eelistad õpetada?

Tauno vastas, et tal ei ole eelistust. Aire aga tõi välja, et tema jaoks on nooremad õpilased jätnud positiivsema mulje. Nooremad õpilased on avatumad ja julgevad rohkem oma mõtteid programmi panna. Samuti teevad nad agaramalt kaasa ja julgevad rohkem katsetada ka võimalusi, millest tunni läbiviija pole rääkinud.

Millised on erinevused tunni andmises väiksemas klassis ning klassis, kus on üle kahekümne õpilase?

Selle küsimuse vastusena tõi Tauno välja, et kui klassis on 30 õpilast ja kui näiteks 3 õpilast räägivad, siis tekitab see juba üsna suure müra. Samuti, mida rohkem on klassis õpilasi, seda keerulisem on kõikide juurde jõuda, eriti kui anda tundi üksinda.

Aire ütles, et lisaks õpilaste arvule mõjutab tunni andmist ka klassis suurus. Juhul, kui õpilasi on vähe, aga klass on suur, siis on Airel ebamugav tundi anda, sest ta hääl on üsna vaikne. Eriti on seda tunda, kui õpilased istuvad nii klassi ees- kui ka tagaosas. Aire arvab samuti, et rohke õpilastearvuga klassis, on keeruline kõikide õpilasteni jõuda. Võib öelda, et Aire ideaalklass oleks väike ja kompaktne, kuna talle meeldiks õpilaste keskel seista.

Millised on erinevused Tartus ja kaugemates kohtades tunni andmisel

Nii Aire kui Tauno arvasid, et kaugemates koolides on õpilaste põnevus suurem ning nende huvi on lihtsam äratada. Lisaks tõi Tauno välja, et kaugemates koolides oli rohkem tunda tunnustust, et inimesed tulevad ülikoolist ja viivad tunni läbi. Sageli kutsus ka direktor endaga rääkima või surus käe pihku. Tartu koolides käiakse päris palju ja sellepärast on ka õpilaste suhtumine pigem ükskõiksem.

Mis oli kõige närvesöövam tegevus koolikülastuste juures?

Aire ei osanud selle küsimuse vastusena midagi välja tuua ja ütles, et kõik tema närvid jäid alles. Tauno jaoks oli kõige närvesöövam kooli kohale jõudmine. Tuli kirjutada kirju ja vaadata, mis teed pidi tuleb sõita. Kõige raskem oli Tauno arvates see, et oli vaja arvestada, kaua kulub aega sõidule ja tihti esines kartus, et jäädakse hiljaks.

Mis oli kõige meeldejäävam?

Kõige meeldejäävam oli Tauno jaoks Valga Vene gümnaasiumi külastamine. Seda sellepärast, et õpilased räägivad vene keeles ning Tauno kartis, kas õpilased ikka saavad aru, mida neile tutvustatakse. Õnneks olid kõik väga sõbralikud ja said tunnis räägitavast aru. Lisaks jäi see koolikülastus Taunole meelde, kuna hiljem kajastati külaskäiku ka meedias.

Tauno ja Aire tõid mõlemad välja selle, et meelde jäid need korrad, kui direktor endaga rääkima kutsus ja tundis huvi, millega koolikülastajad tegelevad ning kuidas neile kool meeldib. Airele jaoks olid kõige meeldejäävamad neljandate ja viiendate klasside õpilased, kuna nad õppisid hästi kiiresti ja oli näha, et neid huvitas tunnis tehtav väga.

Mida andis koolikülastus juurde?

Airele andis programmeerimisalaste koolikülastuste läbiviimine juurde õpetamiskogemust ning samuti sai ta targemaks Pythoni koha pealt, mida ta enne praktiliselt ei osanudki. Tauno sai juurde julgust minna võõrastele inimestele tundi andma.

Kas koolikülastused võiksid olla seotud informaatika õppekavaga?

Nii Aire kui Tauno leiavad, et programmeerimisalased koolikülastused võiksid informaatika õppekavaga seotud olla. Tauno tõi välja, et kui 150 õpilasest käiks iga päev kaks õpilast koolikülastust tegemas, saaksid kõik gümnaasiumid külastatud. Isegi kui käiksid pooled tudengitest, oleks ka juba väga palju, kuna külastusi tehakse üle mitme aasta. Siiski leidis Tauno, et selline programmeerimisalaste koolikülastuste tegemine on ikkagi utoopiline. Ka Aire arvas, et kõik õpilased võiksid käia korra või kaks, kuigi samas leidis ta, et sellest poleks tõenäoliselt kasu, kui kogu aeg käiksid uued inimesed. Seda sellepärast, et alguses on vaja proovida, mis töötab ning siis, kui on leitud sobilik tunnikava, tuleks seda lihvida. Juhul, kui kogu aeg käivad erinevad inimesed, siis on pigem rõhk selles, et tunnid saaksid kuidagi tehtud. Aire arvab, et kui programmeerimisalased koolikülastused õppekavaga siduda, peaks ikkagi olema kindel seltskond, kes sellega tegeleb.

Kui mitu korda koolikülastusi tehes võib külastaja huvi kaotada?

Aire arvab, et koolikülastuste tegija võib huvi kaotada, kui pidevalt käiakse külastusi tegemas kohtades, kuhu sõiduks kulub kaks korda rohkem aega kui kohal ollakse. Samuti tõi Aire välja, et tüdimus võib tekkida, kui näiteks õppeaasta jooksul käiakse kolm päeva nädalas ja iga kord viiakse läbi sama tund. Ise arvab Aire, et temal ei tekkiks tüdimust, kui käia kord nädalas, aga kellelgi teisel võib tekkida. Inimesed on ju erinevad.

Tauno leiab, et kui programmeerimisalaste koolikülastuste eest tasu ei saaks, siis üle kahe või kolme korra käia ei tahaks, sest külastuse jaoks kulub tavaliselt terve päev. Samas aga, kui selle eest saaks ka väikese preemia, siis võiks Tauno arvates koolikülastusi teha aastaringi. Tudengil oleks hea, kui ta saaks sealt väikese boonuse. Nii Tauno kui Aire usuvad, et nad läheksid uuesti koolikülastusi tegema, kui neile ajaliselt sobiks. Samuti sai vähe kasutada lauamänge ning ka töötoad hakkasid just tööle. Lisaks loomulikult tahe tutvustada teistele, millega ülikoolis tegeletakse.

Mida teeksid järgmistel koolikülastustel teisiti?

Tauno tõi välja, et ta täiendaks töölehti pannes töölehe teisele poolele veebilehtede info ja kontaktinfo. Aire arvab, et ta muudaks Pythoni tundi. Selle asemel, et lasta õpilastel lihtsalt ruut ja kolmnurk joonistada, arvab Aire, et ta kasutaks pigem viimast töölehte (vt lisa 4), kus oli ka näidiskood ning õpilased pidid mingit parameetrit muutma, et saada vastav joonis.

Kas oleksid valmis järgmisi koolikülastajaid õpetama? Kuidas sa seda teeksid?

Aire tõi välja, et kodus peaks ikka läbi mõtlema, mida teha tahetakse (kajastatud peatükis 3.1). Tauno korraldaks järgmiste koolikülastajate õpetamiseks kokkusaamise, kus tutvustataks, mida on siiamaani tehtud. Samuti kutsuks ta kohale eelmised koolikülastajad, et nad oma kogemusi jagaksid. Tauno tahaks teada ka, millised on järgmiste koolikülastajate varasemad kogemused ning millise inimesega on tegu, kuna see olevat väga oluline. Lisaks kokkusaamisele, tuleks uued programmeerimisalaste koolikülastuste tegijad vähemalt üks kord kaasa võtta, et nad näeksid, kuidas on siiamaani tunde läbi viidud.

Kokkuvõte

Antud bakalaureusetöö on mõeldud ennekõike materjalina järgmistele programmeerimisalaste koolikülastuste tegijaile. Töö tegemisel tugineti 2013/2014. õppeaastal toimunud programmeerimisalastele koolikülastustele. Koolikülastustel osalesid lisaks bakalaureusetöö autorile ka Aire Kuresson ja Tauno Palts. Viidi läbi 10 koolikülastust, kus tutvustati õpilastele programmeerimiskeeli Scratch ja Python. Samuti räägiti üldisemalt informaatikast ja programmeerimisest ning toodi välja õpilaste ja õpetajate tagasiside koolikülastustele.

Töös seletatakse, miks üldse teha programmeerimisalaseid koolikülastusi. Koolikülastuste käigus laiendati õpilaste silmaringi informaatika valdkonnas ning tutvustati ka sisseastumistingimusi Tartu Ülikooli informaatika erialale. Samuti on programmeerimisalane koolikülastus kasulik ka õpilase seisukohast, kuna nad saavad teada, mis on programmeerimine ning lisaks on pakub selline külastus ka vaheldust tavalisele koolipäevale.

Läbiviidud tundide kohta on olemas näidistöölehed ja tunni läbiviimise kavad. Loomulikult võib järgmiste koolikülastuste tegija vastavaid kavasid muuta või koostada täiesti uued kavad. Näiteks toetusime meie oma sissejuhatuse tegemisel Jaak Vilo loengule, kuid kindlasti on ka teisi võimalusi sissejuhatuse tegemiseks ja antud bakalaureusetöö autor julgustab erinevaid võimalusi katsetama.

Samamoodi ei pea programmeerimiskeeltena tutvustama just Pythonit ja Scratchi. Isegi õpetajad tõid oma tagasisides välja, et oleks tore, kui lisaks Scratchile ja Pythonile pakutaks ka mõne teise programmeerimiskeele tutvustust. Samuti ei pea kasutama lauamängu „*The Robots Game*“, kuid siiski võiks olla midagi, mida saaks läbi viia juhul, kui tund ei toimu arvutiklassis. Meie leidsime lauamängu üsna juhuslikult, aga samas kuna see mäng on juba õpilaste peal katsetatud ning samuti on olemas ka mängulaud ja nupud, siis on seda võib-olla mugavam kasutada, kui hakata ise uut otsima.

Lisaks erinevate töötubade kirjeldusele on bakalaureusetöös välja toodud ka 2013/2014. õppeaastal koolikülastuste tegijate Aire Kuressoni ja Tauno Paltsi muljed ja soovitused järgmistele koolikülastuste tegijatele. Töö autori roll oli kirjutamisel pigem vaatlev ning sellepärast ei ole tema muljeid koolikülastustest antud töös eraldi välja toodud.

Loodetavasti on antud bakalaureusetöös kasulikku infot järgmistele koolikülastuste tegijatele ning siinkohal mainiksin uuesti, et antud töö on ennekõike materjal ja siin kirjutatut ei pea võtma kui reegleid, mida tuleb järgida. Pigem võiksid uued koolikülastuste tegijad katsetada endale meelepäraseid meetodeid ning see võimaldaks tulevikus teha mitmekülgsemaid ja veel huvitavamaid koolikülastusi kui need, mida meie läbi viisime.

Kasutatud kirjandus

[1] Stuart Wray (2013), „*The Robots Game*“ lauamäng, [www] http://onfoodandcoding.blogspot.co.uk/2013/06/the-robots-game.html (08.05.2014)

[2] ProgeTiiger (2014) ProgeTiiger programmi koduleheküIg, [www] http://progetiiger.ee/uudised (07.05.2014)

[3] J. Vilo (2013) „Biti jõud on suur“, [videoloeng] http://www.uttv.ee/naita?id=18203 (08.05.2014)

[4] T. Palts (2013) „Programmeerimisalased koolikülastused“, ettekande slaidid autori valduses (mitteavalik materjal)

[5] Scratch, [www], http://scratch.mit.edu/ Scratchi loomiselehekülg(10.05.2014)

[6] T. Palts (2013) „Spraidi loomine ja tausta muutmine“, [videoloeng] https://www.youtube.com/watch?v=iqEcVSHqDPA (10.05.2014)

[7] M. Resnick, J. Maloney, A. Monroy-Hernandez, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silverman, Y. Kafai. (2009). „Scratch: Programming for All.“, [www] http://cacm.acm.org/magazines/2009/11/48421-Scratch-programming-for-all/fulltext (11.05.2014)

[8] Kidscoder (2014) „Why learn Scratch?“, *Kidscoder* programmi koduleheküIg[www] http://www.kidscoder.com/about_why.html (11.05.2014)

[9] Python(2014) „What is Python?“, Pythoni programmeerimiskeele koduleheküIg [www] https://docs.python.org/2/faq/general.html#what-is-python (11.05.2014)

[10] M. F. Sanner (1999). „Python: a Programming Language for Software Integration and Development“; [pdf] http://www.scripps.edu/sanner/html/papers/NewsAndViewsSept99.pdf (11.05.2014)

[11] Seymour A. Papert (1993) „Mindstorms: Children, Computers, and Powerful Ideas“, [pdf] http://www.arvindguptatoys.com/arvindgupta/mindstorms.pdf (vaadatud 11.05.2014)

[12] Eesti Hariduse Infosüsteem, [www] https://enda.ehis.ee/avalik/avalik/oppeasutus/OppeasutusOtsi.faces (24.04.2014)

[13] K. Helme „Vaata ja võrdle: kui pikad on Euroopas koolivaheajad?“ (28.05.2012) Delfi, [www] http://eesti.info/uudised/uudis.php?uid=1358174 (24.04.2014)

[14] Haridus- ja Teadusministeerium, [www] http://www.hm.ee/index.php?0513744 (11.05.2014)

[15] T. Palts (2014) „Koolikülastused“ ettekande slaidid autori valduses (mitteavalik materjal)

[16] „Programmeerimisalaste koolikülastuste tagasiside vastused“, *Google Drive*’i materjal autori valduses (mitteavalik materjal)

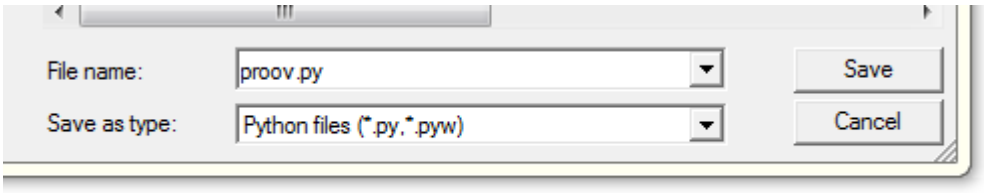
Lisad

Lisa 1. Pythoni tunni tööleht

Pythoni programmeerimine

SISSEJUHATUS

1. Trüki START menüüs otsingusse "IDLE" ja ava IDLE (Python).
2. Trüki tekst "3+4" ja vajuta ENTER. Kas arvutab?
3. Programmi kirjutamiseks tee uus fail (File->New Window).
4. Salvesta fail selliselt, et **faili lõpus oleks ".py",** näiteks: proov.py. Vaata ka pilti:



ESIMENE TESTPROGRAMM

```
from turtle import *


forward(100)
right(90)
forward(100)
```

VEEL KÄSKE

forward(100) – nool liigub 100 pikslit edasi

backward(100) – nool liigub 100 pikslit tagasi

right(90) – nool pöörab 90 kraadi paremale

left(90) – nool pöörab 90 kraadi vasakule

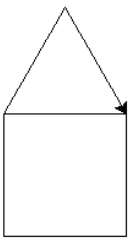
up() – nool enam ei joonista

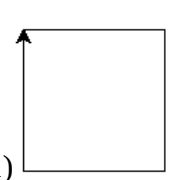
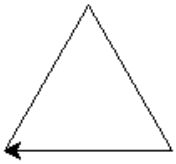
down() – nool hakkab jälle joonistama

speed(1000) – muudab joonistamise kiiremaks

– trelllide järgi olevat koodi ei sooritata. Võib kasutada kommenteerimiseks.

ÜLESANNE 1:



- Joonista: 1)  2)  3)

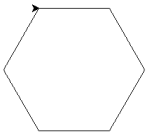
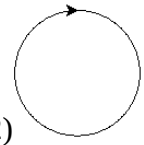
TEINE PROGRAMM

Joonistame ruudu, korrates käske "forward" ja "right" 4 korda:

```
from turtle import *

n = 1
while (n <=4):
    forward(100)
    right(90)
    n = n + 1
```

ÜLESANNE 2

Joonista: 1)  2) 

KOLMAS PROGRAMM

```
from turtle import *
speed(1000)


n = 1
while (n <= 500):
    forward(n)
    right(90)
    n = n + 5
```

ÜLESANNE 2

Katseta viimast programmi muutes erinevaid arve ja proovides, mis siis juhtuma hakkab.

EDU!

autor: Tauno Palts

Lisa 2. Scratchi tunni tööleht

SISU (variant 1)

http://scratch.mit.edu/

Scratchi lühitutvustus:

- keelevahetus;
- logimine: võimaluse tutvustamine; näidistunnis kasutajakontode loomisega tegelema ei hakka (vähemalt tunni alguses mitte);
- töölaud: lava, spraidid ehk tegelased, plokid, skriptid, projekti nimi;
- spraidi lisamise võimalused ja kustutamine;
- tausta lisamise võimalused;

Ülesanne 1

Tee pilt, kus on

- ❑ vähemalt 3 tegelast;
- ❑ vähemalt 1 isejoonistatud tegelane;
- ❑ taustapilt.

- spraidi liikumise programmeerimine;

  Näited:

Ülesanne 2

Täienda eelnevas ülesandes tehtud pilti:

- ❑ lisa tekst;
- ❑ paljunda ühte tegelast;
- ❑ lisa vähemalt ühele tegelasele liikumine.

Täpsem info lühitutvustuse kohta: http://goo.gl/McdoY4

Rohkem eestikeelseid juhendeid Scratchi (ja ka programmeerimiskeele Python) kohta: www.progetiiger.ee

SISU (variant 2)

http://scratch.mit.edu/

Scratchi lühitutvustus:

- keelevahetus;
- logimine: võimaluse tutvustamine; näidistunnis kasutajakontode loomisega tegelema ei hakka (vähemalt tunni alguses mitte);
- töölaud: lava, spraidid ehk tegelased, plokid, skriptid, projekti nimi;
- spraidi lisamise võimalused ja kustutamine;
- tausta lisamise võimalused;
- spraidi liikumise programmeerimine;

  Näited:

- tegevused (reageerimised), kui 2 spraiti kokku puutuvad: kadumine (peitmine), ütlemine, kostüümivahetus;

  Näited:

Ülesanne

Loo mäng, kus on:

- ❏ taustapilt;
- ❏ vähemalt 1 tegelane, kes on juhitav klaviatuurilt;
- ❏ vähemalt 2 tegelast, kes reageerivad erinevalt kokkupuutele teise tegelasega.

Täpsem info lühitutvustuse kohta: http://goo.gl/McdoY4

Rohkem eestikeelseid juhendeid Scratchi (ja ka programmeerimiskeele Python) kohta: www.progetiiger.ee

autor: Aire Kuresson

Lisa 3. Ajakava programmeerimiskeele Python tunni jaoks

AEG	SISU	LISAINFO	MÄRKUSED
Enne tundi	• Avada tühi *Python*i fail, muuta font suuremaks		
10 min	• Tutvustus • Kokkulepe, et sissejuhatuse ajal ei kasuta õpilased arvutit • Sissejuhatus		
5 min	**ALGSEL TUTVUSTUSEL ÄRA NÄITA, KUIDAS *PYTHON*IT AVADA** • Selgitada *Python*i käske *forward(n)*, *backward(n)*, *right(n), left(n)* • Näidata, milline on joonis forward(100) sisestamise järel • Küsida, milline joonis tekib kõrvaloleva koodi sisestamise järel	from turtle import * forward(100) right(90) forward(100)	
25 min	• Näidata, kuidas avada *Python* ning salvestada **(laiend *.py*)** uus fail • Lasta lõpetada alustatud kood, et tekiks ruut • Jagada töölehed • Ülesannete lahendamine		
5 min	• Tagasiside goo.gl/SvqL6n • progetiiger.ee tutvustus		

autor: Aire Kuresson

Lisa 4. Ajakava programmeerimiskeele Scratch tunni jaoks

AEG	SISU	LISAINFO	MÄRKUSED
Enne tundi	•		
10 min	• Tutvustus, lühike sissejuhatus • Scratchi tutvustus I: o scratch.mit.edu o keelevahetus, sisselogimine, erinevad alad o tausta valimine o tegelase valimine, kustutamine o tegelase joonistamine o liikumise programmeerimine: otse, paremale, vasakule o plokkide eemaldamine	• KASS liikuma: *tühik*, →, ← o Sündmused, Liikumine (liigu, pööra paremale, pööra vasakule)	
10 min	• Ülesanne 1: vali taustapilt ning lisa või joonista vähemalt 1 tegelane, kes on juhitav klaviatuurilt		
10 min	• Scratchi tutvustus II: o „oota kuni“ võimalused: 1. **kadumine**: kui kass kohtub VALITUD HIIRega, siis hiir kaob; 2. **ütlemine**: kui kass kohtub JOONISTATUD HIIRega, siis hiir ütleb midagi; o **koopia joonistatud tegelasest** → uuele tegelasele teine kostüüm 3. **kostüümi vahetus** (algselt samad omadused, mis algsel, seega ka programmi algus olemas)	Kui klikkida rohelist lippu, siis mäng algab ja hiired jäävad ootama: 1. VALITUD HIIR kaob: o Sündmused (roh. lipp), Juhtimine (oota kuni), Andurid (puudutab kassi), Välimus (peida) o *Kuidas hiire tagasi saab? Algusesse „näita“.* 2. JOONISTATUD HIIR ütleb: o Sündmused (roh. lipp), Juhtimine (oota kuni), Andurid (puudutab kassi), Välimus (ütle 2s) KOOPIA JOONISTATUD HIIREST → JOONISTA UUS KOSTÜÜM 3. JOONISTATUD HIIR 2 vahetab kostüümi: o *Sündmused (roh. lipp), Juhtimine (oota kuni), Andurid (puudutab kassi),* Välimus (ütle → võta	

		kostüüm2) ○ *Kuidas saada tagasi algse kostüümiga hiir? Algusesse „kostüüm 1“.*	
10 min	• Ülesanne 2: Täienda 1. ülesannet ning loo mäng, kus lisaks eelnevale on vähemalt 2 tegelast, kes reageerivad erinevalt kokkupuutele teisega. *Eelnevalt näidatud reageerimise viisid: ütlemine, kadumine, välimuse muutmine.*		
	• Kui jääb aega: ○ Sündmused (kui vajutatakse klahvi …), Liikumine, Tehted (mine x: juhuarv, mine y: juhuarv)		
5 min	• Kes tahab tehtud töö alles jätta, saab selle salvestada ja endale e-postiga saata • Tagasiside goo.gl/SvqL6n • progetiiger.ee tutvustus		

autor: Aire Kuresson

Lisa 5. Lauamängu töötoa lõpetamine

Millest võiks rääkida lauamängu töötoa lõpus?

- **Kas mäng seostub programmeerimisega? Kui jah, siis kuidas?**

  Peab kirja panema sammsammulise juhise, mida täita. See on nagu väike programm roboti jaoks.

- **Kas mäng seostub kahendsüsteemiga? Kui jah, siis kuidas?**

  0 – ei saa plaanitud (kirjapandud) tegevust teha, 1- saab plaanitud tegevust teha.

- **Mille järgi otsustasite, kuidas oma käike kirja panna?**

  Programmeerimisel on oluline aru saada, mis on eesmärgiks ning otsima võimalusi, kuidas selleni jõuda. Selle mängu korral saab eesmärgiks olla näiteks kiiresti lõppu jõudmine või teiste liikumise takistamine.

- **Kas kirjapandu toimis alati? Kas vahel juhtus, et mõtlesite käike kirja pannes üht, aga kirja panite midagi muud? Kui jah, siis kumba variandi järgi robot liikus?**

Lahenduse peab korralikult läbi mõtlema: kui ütled arvutile (robotile), et peab pöörama paremale, siis nii ka juhtub, kuigi tegelikult mõtlesite vasakule või kui panite kirja EEE, siis robot ei liigu EEEE nagu vaja.

- **Kas suutsite kõike ettetulevat (teised nupud/robotid, müür) arvestada?**
- **Mis te arvate, miks on reeglites kirjas, et iga mänguvooru eel peab kirja panema kuni 10 käiku? Miks ei võiks kirja panna kogu liikumistrajektoori algusest lõppu?**

  Kogu trajektoori kirjapanemisel võivad mõned ettetulevad takistused märkamata jääda. Ka programmeerides on oluline mingi osa koodi kirjutamise järel kontrollida, kas kõik toimib soovitult, seejärel edasi kirjutada ning jälle kontrollida, … Nii on lihtsam vigadest aru saada ja neid parandada. Mängides ju ka tegelikult ei tea, mitmendana teie robot liikuma hakkab ning selleks ajaks võivad tingimused olla muutunud. Kui pikka plaani korraga ei tee, pole kaotus nii suur.

- **Mängu looja lehel** (http://onfoodandcoding.blogspot.co.uk/2013/06/the-robots-game.html): Kuigi arvutid tunduvad olevat targad, pole nad tegelikult targemad kui mängunupud. Nad on lihtsalt kiiremad ja see paneb meid arvama, et nad on targad. Arvutid, nagu ka mängunupud (robotid), ei tee seda, mida me tahame, vaid seda, mida me neile ütleme. Ja kui midagi sai öeldud valesti, tuleb leida võimalus selle parandamiseks. Ka parimad programmeerijad ei suuda alati kõike õigesti kirja panna.

autor: Aire Kuresson

Lisa 6. Lauamängu „The Robots Game“ reeglid [2]

„*The Robots Game*“ reeglid (pisut muudetud võrreldes orginaaliga):

1. Mängu mängitakse kahe võistkonnana. Mängu võidab see võistkond, kes saab oma kolm „robotit“ (nuppu) finishisse.
2. Võistkond koosneb kolmest mängijast ning iga mängija kontrollib ühte „robotit“. „Robotid“ on nummerdatud. Ühel võistkonnal on nupud 1, 2 ja 3 ning teisel võistkonnal on nupud 4, 5 ja 6. Lihtsamaks nupu tuvastamiseks on võistkondade nupud erinevat värvi.
3. „Robotid“ alustavad mängu kuuelt algusruudult. „Robotid“ ei pea olema numbrilises järjekorras. Nooled peaksid olema suunatud mängulaua poole.

4. Iga mängija saab tühja käigulehe (joonis 4), kuhu nad kirjutavad oma „roboti“ käigud. Kindluse mõttes võiks iga mängija käigulehele kirjutada oma „roboti“ numbri.
5. Mäng koosneb käikudest, mida järgitakse niikaua, kuni üks võistkond võidab Üks käik koosneb järgnevast:
    a. Iga mängija kirjutab oma käigulehele „roboti“ jaoks kümme käiku tulpa. Lubatud käigud on **Edasi** (**E**), **Tagasi** (**T**), **Paremale** (**P**) ja **Vasakule** (**V**). Kõiki kümmet käiku ei pea kasutama. Tegelikult ei pea ühtegi käiku tegema, kui ei soovita. Sellisel juhul jääb robot samasse ruutu. Ühe võistkonna mängijatel on lubatud omavahel suhelda ja käike arutada.
    b. Kui kõik mängijad on oma käigud lehele kirja pannud, viskab keegi mängijatest täringut. Vastavalt sellele, mis number tuleb, alustavad mängijad käimist. Näiteks kui täringut veeretades tuleb number 4, käivad mängijad järjekorras 4, 5, 6, 1, 1 ja siis 3. Seega sel hetkel, kui mängijad kirjutavad oma käike, ei tea nad ette, kus teiste mängijate nupud asuvad.
    c. Käikude käimisel loeb nupu omanik käigud ükshaaval ette nii, nagu on nad paberil kirjas. Keegi vastasvõistkonnast liigutab nuppu. Seda sellepärast, et inimesed võivad käike kirja pannes vigu teha. Me tahame aga kindlad olla, et sooritatakse need käigud, mis on lehel kirjas.
    d. Käigud sooritatakse järgnevalt: **Edasi** käsk üritab liigutada „robotit“ ühe ruudu võrra edasi noolega näidatud suunas. See õnnestub, kui eelolev ruut on tühi või saab kasutada „lükkamist“ (vaata punkti 5e). Vastasel juhul käik jääb vahele. Näiteks kui ees on tellissein. **Tagasi** käsk üritab liigutada „robotit“ ühe ruudu võrra tagasi. Käigu õnnestumine toimib samade reeglite järgi nagu Edasi käsu puhul. Käsud **Parem** ja **Vasak** keeravad robotid kohapeal 90 kraadi kas vasakule või paremale. Mõlemad käigud õnnestuvad alati.
    e. „Lükkamine“. Kui „robot“ üritab liikuda ruutu, kus on juba teine „robot“, siis see õnnestub kui „robotil on hoog sees“ ehk kui „roboti“ ja lükatava roboti vahel oli enne üks ruut vahel ja teisel pool lükatavat robotid on tühi ruut. Kui lükkamine õnnestub, liigub lükatav robot ühe ruudu võrra lükatavas suunas edasi (mitte suunas, kuhu näitab roboti peal olev nool). Robot, mis sooritas käigu, hõivab ruudu, milles oli enne teine robot. käigu jooksul saab sooritada maksimaalselt kaks „lükkamist“. Kusjuures eelmainitud tingimused peavad kehtima. Seega ei ole võimalik ühte robotit kahe ruudu võrra edasi lükata.

f. Kõik „roboti“ käigud sooritatakse selles järjekorras, kuidas nad on kirja pandud, olenemata sellest, kas eelmine käik õnnestus või mitte.

6. Mänug lõpp. Laua ühes otsas olevad neli ruutu, milles on igasühes mängulauast väljapoole suunatud nool, on finishiruudud. Mängu võidab võistkond, kes jõuab kõigi oma kolme nupuga finishisse. Juba finishis olevaid nuppe ei ole võimalik lükata ning nende poolt hõivatud ruutudes ei ole võimalik lõpetada.

Lisa 7. Ajakava lauamängu tunni jaoks

AEG	SISU	LISAINFO	MÄRKUSED
Enne tundi	• Mängulauad ja nupud valmis panna		
10 – 15 min	• Tutvustus • Õpilased peavad jagunema 6-liikmelistesse rühmadesse ja iga rühm kaheks võistkonnaks • Mängureeglite tutvustus **(NB! parem-vasak)** • Kokkulepe, et kui tunni lõpuni on 10 minutit, siis lõpetatakse ring ja seejärel mäng katkeb	Iga rühm otsustab ise, • kuidas 2 võistkonda moodustada; • millises järjekorras mängunupud stardijoonele asetada.	
20 – 25 min	• Mängimine	• Meelde tuletada, et teised liikmed ka jälgiksid ütleja ja liigutaja tegevuste kokkulangevust	
7 min	• Kokkuvõte • progetiiger.ee tutvustus	• Üks mäng jääb koolile, seda võib ise edasi arendada muutes või täiendades reegleid. Reeglite lehel ka viide, kust mäng pärineb	
3 min	• Tagasiside		

autor: Aire Kuresson

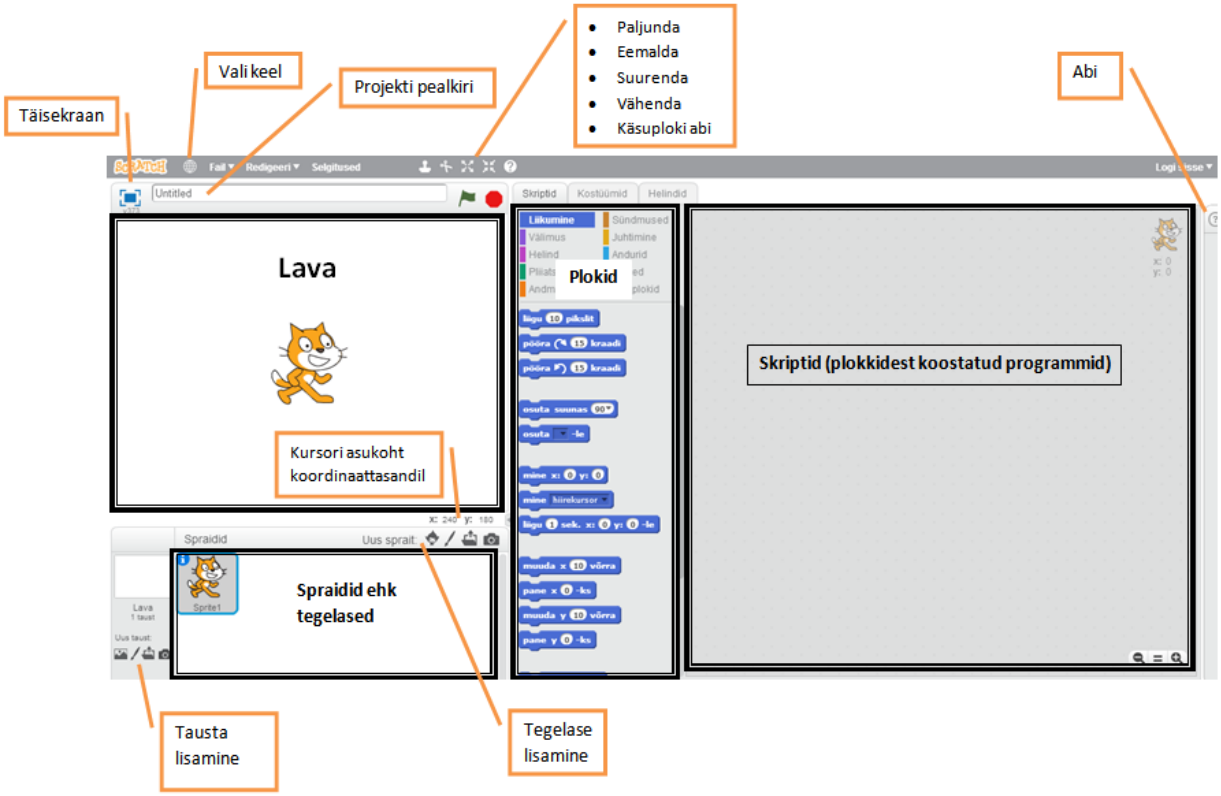
Lisa 8. Õpilastele tutvustatavad Scratchi elemendid

Sissejuhatus

Scratchi kasutamiseks mine leheküljele http://scratch.mit.edu/, vajuta lehe ülemises parempoolses servas **Ühine Scratchiga** (kui Scratchi kontot veel pole) või **Logi sisse** (kui Scratchi konto on juba olemas) ning loo endale Scratchi konto või logi sisse olemasoleva kasutajanime ja parooliga.

Et need tekstid oleksid eestikeelsed, peab lehe alumises parempoolses servas olema valitud **Eesti**.

Uue Scratchi projekti tegemiseks vajuta pärast sisselogimist lehe ülemises servas **Loomine.** Seejärel avaneb lehekülg, kus on näha üks sprait ehk tegelane:



Tekkinud faili nimi on näha lehe ülemises vasakpoolses servas (sõna Untitled, mille järel võib olla ka arv), selle asemele võid kirjutada uue faili nime ning seejärel valida **Fail**-menüüst **Salvesta nüüd.** Sama menüü kaudu saab luua ka uue projekti.

http://www.youtube.com/watch?v=oEge5zplm2s

Spraidi ja tausta lisamine

Spraidi lisamine

Uue spraidi lisamiseks on 4 võimalust (vastavad ikoonid ):

1. Vali sprait teegist, s.t. vali oma projekti uus sprait olemasolevate tegelaste hulgast. Selleks vajuta esimesele ikoonile, otsi sobiv tegelane ning tee sellel hiire topeltklikk.
2. Uue spraidi joonistamine

   Klikkides sellel nupul, tekib lehe paremale poolele ala, kus saab joonistada. Vahendid on näha selle ala vasakpoolses servas:

    endale sobiva kujuga joone tegemine

    sirge joone lisamine

    ristkülik; kui valid selle kujundi ning hoiad enne joonistamist all *Shift*-klahvi, saad teha ruudu

    ellips; kui valid selle kujundi ning hoiad enne joonistamist all *Shift*-klahvi, saad teha ringi

    teksti sisestamine

    joontega piiratud ala täitmine värviga

    Kustutamine

    tehtud joonistuse piiramine raamiga, et muuta selle suurust või asukohta

    tehtud joonistuse piiramine raamiga, et seda kopeerida teise kohta samal joonistusel (tõmba raam ümber selle osa, mida tahad kopeerida, mine raami keskele, et tekiks käe kujutis ning liiguta raamiga ala vajalikku kohta)

   Lisaks on joonistamise ala all võimalik valida joonistusvahendi, kujundi või teksti värvi (vali värv enne kui hakkad valitud kujundit joonistama) ning olenevalt tehtud valikust ka kirjastiili, joone paksust, kustutaja laiust või kas kujund on kohe joonistades seest värvitud või tühi.
3. Spraidi laadimine failist, s.t. otsi sobiv sprait arvuti pildifailide hulgast.
4. Uus sprait veebikaameraga

Spraidi paljundamine

Kui soovid oma projekti mitut ühesugust spraiti, siis tee olemasoleval spraidil hiire paremklikk ning vali tekkinud menüüst **paljunda.**

Spraidi kustutamine

Kui soovid mõne spraidi kustutada, siis tee olemasoleval spraidil hiire paremklikk ning vali tekkinud menüüst **eemalda.**

Tausta lisamine

Tausta lisamiseks on samuti 4 võimalust (vastavad ikoonid ):

1. Vali taust teegist, s.t. vali oma projekti taust olemasolevate taustade hulgast. Selleks vajuta esimesele ikoonile, otsi sobiv taust ning tee sellel hiire topeltklikk.
2. Uue spraidi joonistamine - see on sarnane spraidi joonistamisega.
3. Tausta laadimine failist, s.t. otsi sobiv taust arvuti pildifailide hulgast.
4. Uus taust veebikaameraga

autor: Aire Kuresson (koostatud videoloengupõhjal [6])

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina **Heike Randlahe** (sünnikuupäev: 10.11.1991)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose **Programmeerimisalased koolikülastused**, mille juhendaja on Eno Tõnisson,

   1.1 reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

   1.2 üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **16.05.2014**